COMPUTER-AIDED-DESIGN AND DISPLAY OF THREE-DIMENSIONAL SURFACES

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

by

D. GANGULY

to the

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

JANUARY, 1988

ME-1988-M-GAG-COM-

18/1/88

CERTIFICATE

This is to certify that the thesis entitled "COMPUTER-AIDED-DESIGN AND DISPLAY OF THREE-DIMENSIONAL SURFACES", by D. GANGULY, is a bonafide record of work done by him under my guidance and supervision, for the award of the degree of Master of Technology at the Indian Institute of Technology, Kanpur. The work carried out in this thesis has not been submitted elsewhere for the award of a degree.

Dhande

(Dr. SANJAY G DHANDE)
Professor,
Dept. of Mech. Engg.
&
Dept. of Comp. Sci.
I.I.T., Kanpur.

ACKNOWLEDGEMENTS

I acknowledge with sincerity and gratitude, the guidance provided by Dr. S. G. Dhande in introducing me to the subject of surface design and sorting out various problems through the course of this thesis. I also thank him for his contributions made by way of advices and criticisms at different stages of this work.

I thank the CAD-Project staff at IIT, Kanpur for providing excellent facilities in the form of the NORSK DATA computer system and colour graphic terminals.

D. GANGULY

JANUARY 18, 1988
I.I.T. KANPUR

CONTENTS

LIST OF FIGURES

NOMENCLATURE

$\underline{a}$	Vector defining x-axis of a cutting plane
$\underline{b}$	Vector defining y-axis of a cutting plane
B	Blending functions
d	Distance of a plane from the origin
J	Bezier blending functions
k,l	Order of B-spline surface or curve
m	Number of control points in u-direction
n	Number of control points in v-direction
N	B-spline blending functions
$\underline{n}$	Normal vector to a surface at a point
$\underline{p}$	Vectors defining a plane
$\underline{r}$	Vector defining control points of a surface or curve
$\underline{r}'$	Slope vector of a curve or surface
$\underline{r}(w)$	Vectors defining a parametric curve
$\underline{r}(u,v)$	Vectors defining a parametric surface
$\underline{t}$	Tangent plane to a surface
t,u,v,w	Parameters for parametric curve or a surface
w(i)	Weights

ABSTRACT

In the present work, an attempt has been made to use mathematical formulations of surfaces in computer-aided-design of three-dimensional free-form sculptured surfaces. A software package, SURF-3D, has been developed and installed using spline, Bezier and B-Spline representations. Spline representations are based on the classical 'lofting' techniques. Bezier and B-Spline schemes are easily adaptable to interactive design environments because of their inherent properties of ease in definition and manipulation. The package helps in designing and displaying surfaces of various types. These schemes help in designing axi-symmetric components also. In addition, alogorithms for viewing designed surfaces in parallel or perspective projections, for intersections of surfaces with planes and for generating hidden surface and hidden line displays have also been devised. The package has been tested using several illustrative examples.

CHAPTER I

INTRODUCTION

1.1 Surfaces in Engineering.

Surfaces and surface representations play a vital role in most engineering design and manufacturing processes. The manufacture of aesthetically and aerodynamically designed automobile and aircraft bodies, hull design of ships, machine parts, design of glassware, clothing, shoes etc. are obvious examples.

For all such surfaces, the present practice of design is based on the use of templates [3]. Templates are nothing but sheets cut according to a particular contour obtainable from a surface when it is cut by a plane at a particular position. An ensemble of such templates collectively describes the nature of the surface envisaged.

There are obvious disadvantages to such an approach for design and manufacture of surfaces. The overall procedure of design, manufacture and modification is time consuming. Visual inspection is a must for such surfaces and therefore results in costly prototyping.

With the recent advances in Computational Geometry

concepts, it is now possible to mathematically represent such surfaces [3,1]. Closed form mathematical equations for regular surfaces, such as conics, paraboloids, hyperboloids etc. were available even earlier. Use of such concepts has given credence to representation of even free-form sculptured surfaces.

Computer Graphics has come in to play a vital role in this respect. Now surfaces can be visualized in their proper perspective before manufacture. Analyses of various types can be carried out on the mathematical models of such surfaces. It has, therefore, proved to be a 'pre-service' test.

Generally surface representations have two different uses, 'analytic' and 'synthetic' [8].

Analytic uses require a precise 'fit' of the surface to the data points on the surface. Examples of these could be turbine blades, aerofoil sections of aircraft, etc.

Synthetic use of such representations lends itself to creation, modification and display of a model of a shape interactively, and examination and improvement of the design until it is acceptable. The resulting surface may either be used to generate a new image or for generating numerically controlled machine instructions for actual manufacturing of the surface [8].

1.2 Review of developments.

Various representational schemes have been formulated and studied to properly model such surfaces. Earliest among these are the classical 'lofting' techniques of the ship-building, automobile and aircraft industries [10]. A Spline surface representation was generally used for lofting.

In the mid 1960s S.A. Coons (1963) of MIT and J.C. Ferguson (1964) of Boeing Corporation began work in sculptured surfaces [1]. At about this time, General Motors developed their DAC-1 system [7]. Several other companies, notably Lockheed and McDonnell Douglas also made significant developments [7].

Both Coons and Ferguson used parametric cubics for their surface representations. Bezier (1972) evolved a successful surface design system UNISURF, which has been used in the Renault Co. of France for the design of automobile bodies [2]. In the mid 1970s R.F. Riesenfield used B-Splines for representation of surfaces for the first time [1]. Their emphasis was on the accurate representation of surfaces and on interpolating techniques to meet a variety of design criteria.

The different representational schemes have been primarily described in literature on differential geometry, computational geometry and computer graphics. Several excellent titles such as Faux and Pratt [3], Rogers and Adams [10], Mortenson [7] describe the formulations for

design of different types of surfaces. An excellent treatise on practical use of surface design and manufacturing is by Bezier [2].

State-of-the-art techniques in design and display of surfaces can be found in Gregory [5] and Barnhill and Riesenfield [1].

Viewing and display of surfaces, including shading and rendering can be found in Newman and Sproull [8], Foley and Van Dam [4], Rogers and Adams [10] and Rogers [9]. Useful algorithms for viewing and display can also be obtained from Harrington [6].

1.3 Surface design methodology.

In this thesis, the treatment of the mathematical techniques applied to modelling of shape is primarily concentrated upon the synthetic approach. This approach has been adopted because it helps in guiding the designer to go through the stages of design and display in a more logical and intuitive manner than the analytic approach. Furthermore most synthetic methods can be easily applied to analytic fitting problems (especially for fitting smooth surfaces), whereas analytical methods are not easily amenable to problems of free-form and sculptured surfaces.

The modelling system, therefore can support a class of shapes that is matched to the designers' requirements. It is

possible to model simple mechanical component shapes, or on the other hand, to design as complex a shape as an automobile body or an aircraft fuselage.

The design process in the program is interactive, i.e. the designer first visualizes a shape, then applies modifications to certain parameters to change the shape, until he is satisfied with the design.

The first step, is therefore, the generation of images of shapes that have sufficient realism to communicate to the designer the shape information needed to evaluate the design.

The second step is difficult, since the designer has to express the required changes in a form which is supported by the modelling technique. It has been kept in mind that the modifications required by the designer should be expressed in such a way that a minimal number of parameters need to be changed to achieve the desired result. Here, it is required that the designer is sufficiently familiar with the shape representation to be able to identify which parameters to change and by how much to anticipate very accurately the shape changes that will result.

This last step of selecting parameters to change, illustrates the problem of selecting a shape representation: if only a small set of parameters is provided, then the designers' task is simplified, but the range of shapes that can be represented is likely to be limited. If there are

many parameters, the designer can achieve more varied and more accurate models of his design, but may be overwhelmed by the flexibility offered.

Consequently, keeping these in view, shape representations have been chosen which allow a variable number of parameters: the designer need use as many or as few as necessary to create an acceptable model.

Mathematically tractable and computationally efficient formulation schemes have been adopted in this thesis. These representational schemes come close to matching the designers' 'feel' for design and modification of surfaces.

The above considerations combine to outline the properties of the mathematical representations used :-

(i) Control points :

One of the most common ways of controlling the shape of a surface interactively is to specify points through which the surface must be fitted or that which control the surface shape in a predictable way.

(ii) Multiple values :

A three-dimensional surface is usually not a single-valued function of two coordinates. The surface may loop over itself.

(iii) Axes independence :

The shape of the surface should be same, no matter what coordinate system is used. If the control points are rotated then the surface should rotate by the same amount without change in its shape.

(iv) Global or Local control :

When a control point is changed, the shape of the surface may be altered only in the region of the control point(local), or it may change the shape throughout(global).

(v) Variation diminishing property :

Some mathematical representations have a tendency to magnify rather than smooth, small irregularities in the shape outlined by the control points. Others, possessing a variation-diminishing property, always smooth the designers' control points.

(vi) Versatility :

A shape representation must be flexible enough to enable a designer to model a variety of shapes using control points as tools.

With these requirements in view, it was decided to include only those representations which fulfil these criteria.

All the representations used in this thesis are based upon the parametric, vector-valued function. A point on a

surface is represented as a vector -- a function of two parameters u and v.(See figure 1.1)

$$\underline{r}(u,v) = [\ x(u,v)\ y(u,v)\ z(u,v)\ ]^T \qquad 1.3.1$$

As the parameters take on values, usually between zero and one, the parametric functions trace out a curve on a surface.

The following three mathematical representations have been implemented :-

(1) Spline blended surfaces
(2) Bezier surfaces
(3) B-Spline surfaces.

The Spline representation fits a smooth surface which passes through the control points. Surfaces, which are presently in use may be input using this technique. Modification to a control point in this scheme results in unpredictable changes, and therefore not suitable for use for changes in shape.

Bezier surfaces, on the other hand only approximate the control points. Modifications can be made in an intuitive manner. However, a change made to a control point results in the alteration of all or major part of the surface, thereby making it unsuitable for fine adjustments.

B-Spline surfaces behave similar to Bezier surfaces. The surface approximates the control points closer than a

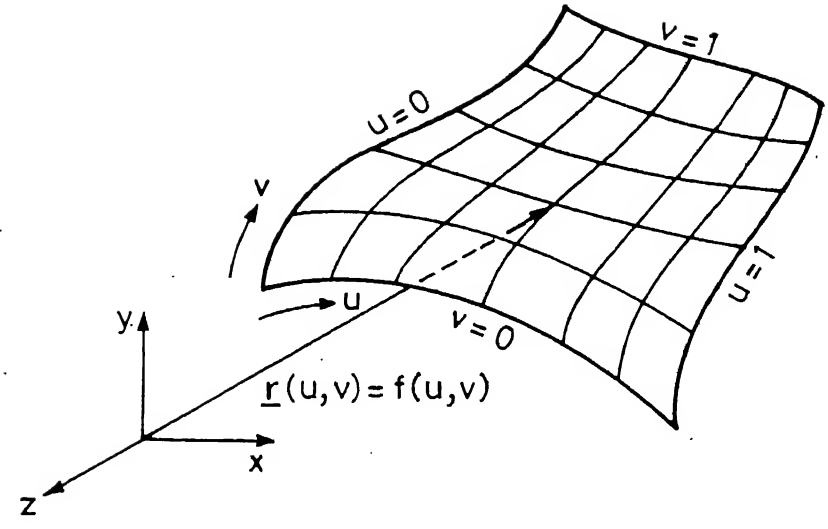


Fig.1.1 A parametric surface with u & v family of curves.

Bezier surface. Modifications to a B-Spline surface are local in nature.

Facilities to convert from one surface type to another has been provided for different types of modification requirements.

Intersections with arbitrary planes may also be visualized by invoking the intersection commands.

Display facilities have been provided which enables the designer to view the surface in any orientation. Parallel and perspective projection transformations have been included in the viewing operations, which the user can manipulate.

CHAPTER II

PARAMETRIC CURVES AND SURFACES

2.1 Differential Geometry concepts in curve and surface design.

A surface defined as a bi-parametric equation is of the form

$$\underline{r}(u,v) = [\ x(u,v)\ y(u,v)\ z(u,v)\ ]^T \qquad 2.1.1$$

where u and v are the parameters and $\underline{r}$ defines the vector to the surface, where

$$\underline{r} = \underline{r}(u,v) \qquad 2.1.2$$

A normal vector $\underline{n}(u,v)$ to a surface at a point defined by $\underline{r}(u,v)$ is given by

$$\underline{n} = \frac{\frac{\partial \underline{r}}{\partial u} \times \frac{\partial \underline{r}}{\partial v}}{\left| \frac{\partial \underline{r}}{\partial u} \times \frac{\partial \underline{r}}{\partial v} \right|} \qquad 2.1.3$$

where $\underline{n}$ represents the unit normal vector as a result of the vector product of the tangent vectors $\partial \underline{r}/\partial u$ and $\partial \underline{r}/\partial v$ at the point. The order in which the vector product is taken, determines the direction of $\underline{n}$. (See figure 2.1.1)

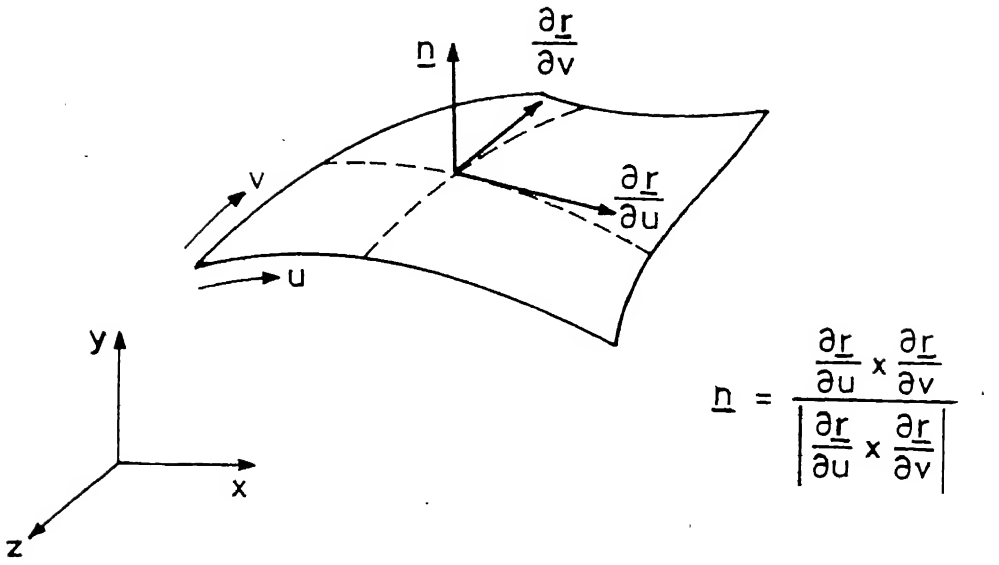


Fig. 2.1.1 Normal vector to a surface.

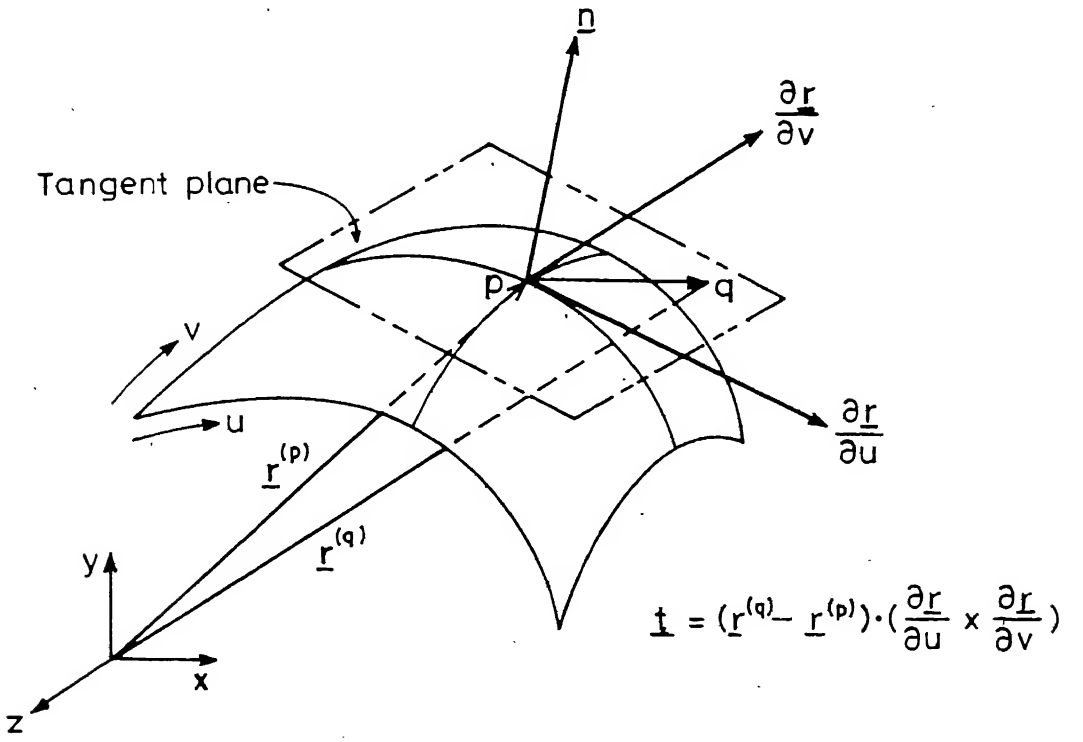


Fig. 2.1.2 Tangent plane.

The equation of $\underline{t}$, a plane tangent to a surface at a point $\underline{r}$ is given by

$$\underline{t} = \left(\underline{r}^q - \underline{r}^p\right).\left(\frac{\delta \underline{r}}{\delta u} \times \frac{\delta \underline{r}}{\delta v}\right) = 0 \qquad 2.1.4$$

where $\underline{r}^q$ is a vector defining a point on the tangent plane, $\underline{r}^p$ is the vector defining the point p on the surface at which the plane is tangent. (See figure 2.1.2)

2.2 Spline curve and surface.

The equation for a single parametric cubic spline segment, in terms of a parametric variable w, is given by

$$\underline{r}(w) = \sum_{i=1}^{4} B_i w^{i-1} \qquad 0 \leq w \leq 1 \qquad 2.2.1$$

where $\underline{r}(w) = [\, x(w)\ y(w)\ z(w)\,]^T$. $\underline{r}(w)$ may be considered as the position vector of any point on the spline segment. [10]. The coefficients B_i are determined by specifying four boundary conditions for the spline segment. (See figure 2.2.1).

The required boundary conditions for each cubic segment consist of two end-points and the tangent vector at each end-point. For a single segment between $\underline{r}_1$ and $\underline{r}_2$, these conditions are given by

$$\underline{r}(0) = \underline{r}_1$$

$$\underline{r}(1) = \underline{r}_2$$

$$\frac{\partial \underline{r}}{\partial w}\Big|_{w=0} = \underline{r}'_1 \qquad 2.2.2$$

$$\frac{\partial \underline{r}}{\partial w}\Big|_{w=1} = \underline{r}'_2$$

Therefore the parametric spline segment, making use of the boundary conditions, is defined as

$$\underline{r}(w) = \underline{r}_1(1-3w^2+2w^3) + \underline{r}_2(3w^2-2w^3) + \underline{r}'_1(w-2w^2+w^3) + \underline{r}'_2(-w^2+w^3) \qquad 2.2.3$$

Now, taking a surface represented by two parameters u and v; a spline surface which satisfies the boundary conditions along the u edges is given by

$$\underline{r}(u,v) = \underline{r}(0,v)(1-3u^2+2u^3) + \underline{r}(1,v)(3u^2-2u^3) + \frac{\delta \underline{r}}{\delta u}(0,w).(u-2u^2+u^3) + \frac{\delta \underline{r}}{\delta v}(1,v).(-u^2+u^3) \qquad 2.2.4$$

Similarly a spline surface, which satisfies, the boundary conditions along w edges is given by

$$\underline{r}(u,v) = \underline{r}(u,0).(1-3v^2+2v^3) + \underline{r}(u,1).(3v^2-2v^3) + \frac{\delta \underline{r}}{\delta v}(u,0).(v-2v^2+v^3) + \frac{\delta \underline{r}}{\delta v}(u,1).(-v^2+v^3) \qquad 2.2.5$$

Figure 2.2.2 depicts a spline blended surface.

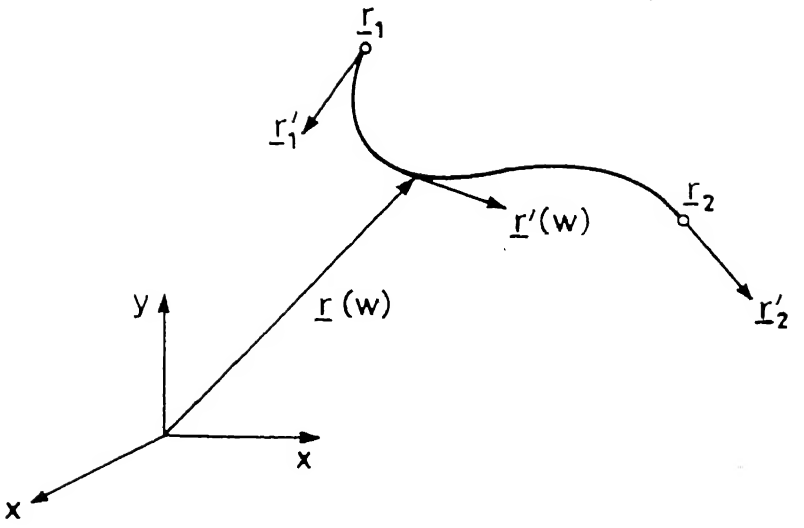


Fig. 2.2.1 Elements of a parametric spline curve segment .

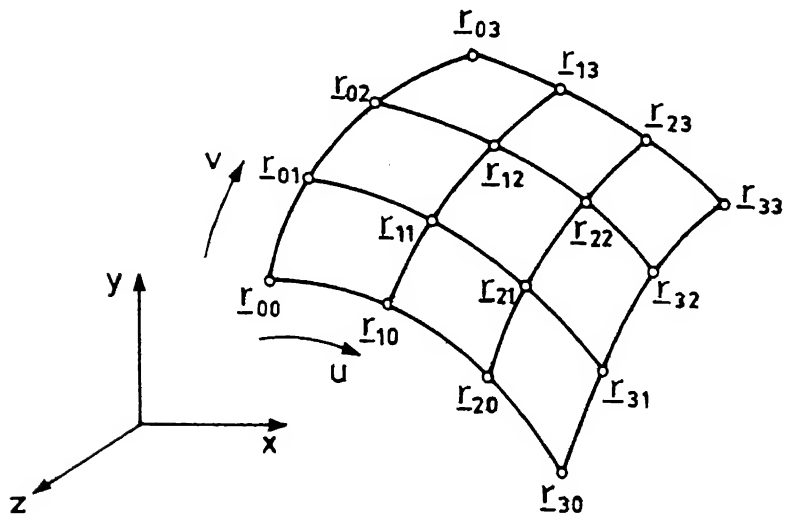


Fig. 2.2.2 Spline blended surface

2.3 Bezier curve and surface.

The mathematical basis of the Bezier curve is a polynomial blending function which interpolates between the first and last vertices of a characteristic polygon defining the curve. The Bezier polynomial is related to the Bernstein polynomial. Thus, the Bezier curve is said to have a Bernstein basis. The basis function is given by

$$J_{n,i}(w) = {}^{n}C_{i}.w^{i}.(1-w)^{n-i} \qquad 2.3.1$$

$${}^{n}C_{i} = \frac{n!}{i!(n-i)!} \qquad 2.3.2$$

with n being the degree of the polynomial and i the particular vertex in the ordered set $\underline{r}_0$ to $\underline{r}_n$. In general an nth order polynomial is specified by n+1 vertices. The points on the curve are then given by

$$\underline{r}(w) = \sum_{i=0}^{n} \underline{r}_i J_{n,i}(w) \qquad 0 \leq w \leq 1 \qquad 2.3.3$$

The surface equation for a Bezier surface involves two parameters u and v and is algebraically defined as

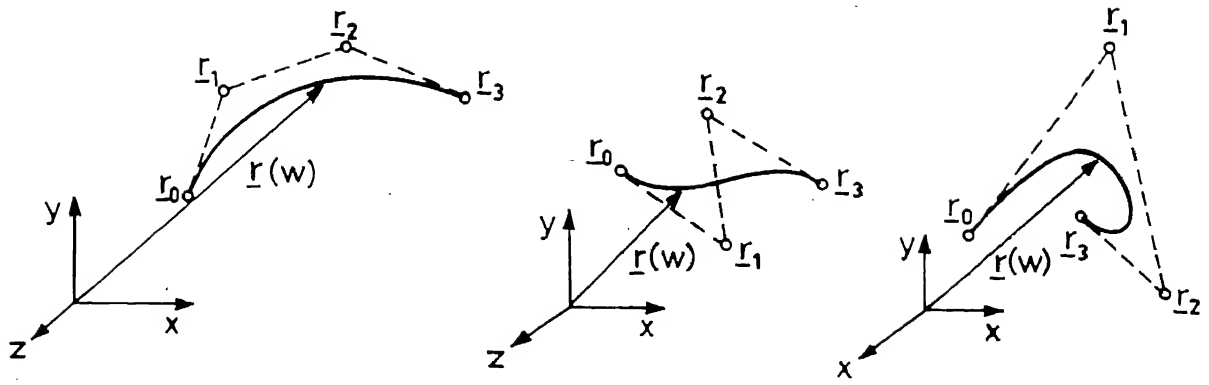


Fig. 2.3.1 Bezier curves with their control polygons.

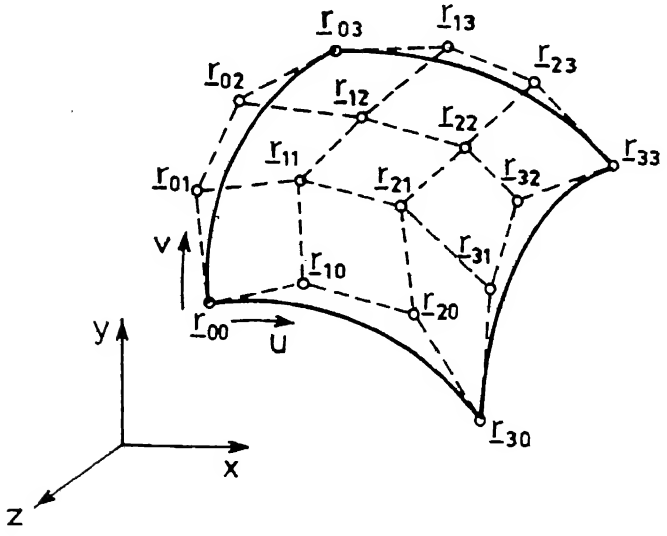


Fig. 2.3.2 Bezier surface with its control polyhedron.

$$\underline{r}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} \underline{r}_{ij} J_{m,i}(u).J_{n,j}(v) \qquad 0 \le u,v \le 1 \qquad 2.3.4$$

See figures 2.3.1 and 2.3.2 for Bezier curve and surface.

2.4 B-Spline curve and surface.

B-spline curves are similar to Bezier curves in that a set of blending functions combines the effects of n+1 control points $\underline{r}_i$. The order of these polynomial blending functions is specially controlled by a parameter k. The b_spline basis functions are recursively defined as

$$N_{i,1} = 1 \qquad \text{if } t_i \le w < t_{i+1}$$
$$= 0 \qquad \text{otherwise} \qquad 2.4.1$$

and

$$N_{i,k}(u) = \frac{(w-t_i)N_{i,k-1}(w)}{t_{i+k-1}-t_i} + \frac{(t_{i+k}-w)N_{i+1,k-1}(w)}{t_{i+k}-t_{i+1}} \qquad 2.4.2$$

where k controls the order of the resulting basis polynomials in the parameter w and thus also controls the continuity of the curve.

The t_i are called knot values and they relate the parameter w to the $\underline{r}_i$ control points.

The points on a b-spline curve are then given by

$$\underline{r}(w) = \sum_{i=0}^{n} \underline{r}_i N_{i,k}(w) \qquad 2.4.3$$

where

$$\begin{aligned} t_i &= 0 && \text{if } i < k \\ t_i &= i-k+1 && \text{if } k \leq i \leq n \\ t_i &= n-k+2 && \text{if } i > n \end{aligned} \qquad 2.4.4$$

with $0 \leq i \leq n+k$ 2.4.5

and $0 \leq w \leq n-k+2$ 2.4.6

For b-spline surfaces, extending equation (2.4.3) to take two parameters u and v,

$$\underline{r}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \underline{r}_{ij} N_{i,k}(u).N_{j,l}(v) \qquad 2.4.7$$

where m and n define the control net of points and k and l indicate the orders of the polynomials in the u and v directions respectively. See figures 2.4.1 and 2.4.2.

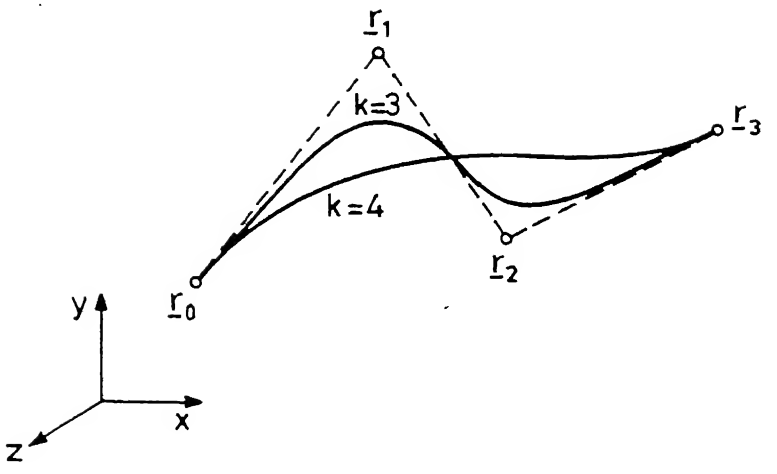


Fig. 2.4.1 3rd and 4th order B-Spline curves with the control polygon.

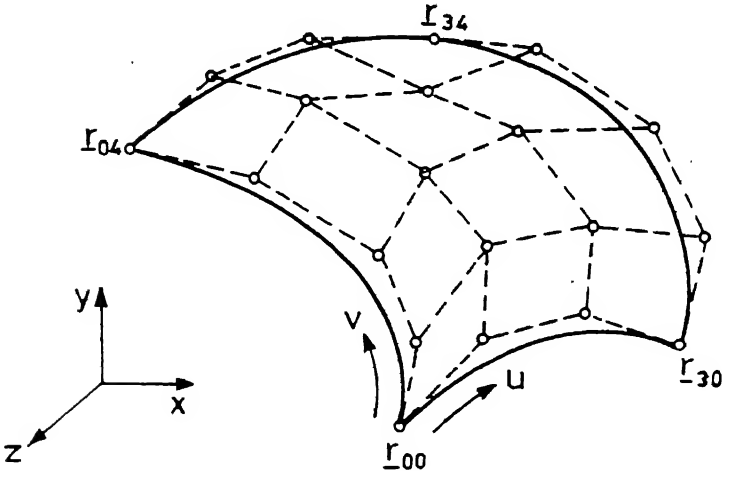


Fig. 2.4.2 A Bicubic B-Spline surface with its control polyhedron.

2.5 Rational parametric curves and surfaces.

There are certain advantages to be gained by defining curves and surfaces in terms of functions more general than ordinary polynomials as used in the formulations described in the earlier sections.

Conic sections can only be approximated by the polynomial blending functions used in spline, Bezier and b-spline formulations. Rational parametric functions allow exact representations of such curves and surfaces.

It has been shown that a parametric polynomial curve of degree n can be written in terms of its basis function $B_i(t)$ as (see equations 2.3.3 and 2.4.3)

$$\underline{r}(t) = \sum_{i=0}^{n} \underline{r}_i B_i(t) \qquad 2.5.1$$

More generally, a rational curve may be defined by

$$\underline{r}(t) = \frac{\sum_{i=0}^{n} \underline{r}_i B_i(t)}{\sum_{i=0}^{n} w_i B_i(t)} = \frac{1}{w(t)} \sum_{i=0}^{n} \underline{r}_i w_i B_i(t) \qquad 2.5.2$$

where t is the parametric variable and

$$w(t) = \sum_{i=0}^{n} w_i B_i(t) \qquad 2.5.3$$

and w_i are scalar weights. Note that if the $B_i(t)$ sum to one (as is the case with spline and Bezier curve segments), the above equation reduces to simple polynomial form. The rational form is therefore more general; the weights may be manipulated to gain greater freedom in curve and surface definition. The rational form can be re-expressed as a polynomial in terms of four-dimensional vectors using homogeneous coordinates. On multiplying throughout by $w(t)$ equation (2.5.2) and writing ξ_i , η_i and ζ_i for the components of $\underline{r}_i$, the component equations

$$\begin{aligned} w(t)x(t) &= \sum \xi_i w_i B_i(t) \\ w(t)y(t) &= \sum \eta_i w_i B_i(t) \\ w(t)z(t) &= \sum \zeta_i w_i B_i(t) \end{aligned} \qquad 2.5.4$$

when taken together with equation (2.5.3), these equations may be summarized as

$$\underline{R}(t) = \sum_{i=0}^{n} \underline{R}_i B_i(t) \qquad 2.5.5$$

in which

$$\underline{R}_i = \left[\begin{array}{cccc} w_i\xi_i & w_i\eta_i & w_i\zeta_i & w_i \end{array} \right]^T \qquad 2.5.6$$

and

$$\underline{R} = [\ wx\ wy\ wz\ w\ ]^T \qquad 2.5.7$$

Rational parametric surfaces require a similar extension using two parametric variables u and v. Further discussion is given in Faux and Pratt [3].

2.6 Bezier control points of a surface.

Starting from a spline-blended surface 'fitted' over a m x n grid of control points, it is possible to generate the Bezier control points which interpolates the same surface.

The advantage of using this technique is mainly to help in later modification of the surface in a much more intuitive and easier method than by changing the positions of the surface points themselves.

Parametric spline - blended surfaces have their parametric intervals $0 \leq u \leq 1$ and $0 \leq w \leq 1$ for each patch on the surface, whereas Bezier surfaces have the same interval over the entire surface. Therefore reparametrization of the spline-blended surface is required. Since the number of points in the u-direction is m+1, an obvious choice for the stepsize of the parametric variable u is 1/m. Similarly in the v-direction the stepsize is 1/n.

A Bezier surface is as given in equation (2.3.4), where $\underline{r}(u,v)$ is the resulting surface. Rewriting equation (2.3.4)

in the matrix form

$$[J(u)].[\underline{r}].[J(v)] = [\underline{r}(u,v)] \qquad 2.6.1$$

where $[\underline{r}]$ is the unknown matrix of control points for the Bezier surface, and $[J(u)]$ and $[J(v)]$ are the square matrices of blending function values in the u and v-directions respectively, of the Bezier surface.

$[J(u)]$ and $[J(v)]$ are positive definite, and hence their inverses exist. Premultiplying equation (2.6.1) by $[J(u)]^{-1}$

$$[\underline{r}].[J(v)] = [J(u)]^{-1}.[\underline{r}(u,v)] \qquad 2.6.2$$

Transposing both sides

$$[J(v)]^{T}.[\underline{r}]^{T} = [\underline{r}(u,v)]^{T}.[[J(u)]^{-1}]^{T} \qquad 2.6.3$$

premultiplying both sides by $[[J(v)]^{T}]^{-1}$

$$[\underline{r}]^{T} = [[J(v)]^{T}]^{-1}.[\underline{r}(u,v)].[[J(u)]^{-1}]^{T} \qquad 2.6.4$$

or

$$[\underline{r}] = [J(u)]^{-1}.[\underline{r}(u,v)].[[[J(v)]^{T}]^{-1}]^{T} \qquad 2.6.5$$

where $[\underline{r}]$ is the required matrix of Bezier control points of the surface defined by $[\underline{r}(u,v)]$.

2.7 B-Spline control points of a surface.

A procedure similar to that described in section (2.6) for Bezier surfaces may be used to obtain the b-spline control points of a given surface.

A b-spline surface is defined by equation (2.4.7). Rewriting this equation in the matrix form

$$[N(u)].[\underline{r}].[N(v)] = [\underline{r}(u,v)] \qquad 2.7.1$$

following the same procedure as used in section (2.6)

$$[\underline{r}] = [N(u)]^{-1}.[\underline{r}(u,v)].[[[N(v)]^{T}]^{-1}]^{T} \qquad 2.7.2$$

where $[\underline{r}]$ is the required matrix of b-spline control vertices.

2.8 Intersections of surfaces with arbitrary planes.

A heuristic approach has been adopted to get the intersection curve between a surface and a plane. The overall procedure has been outlined below :

(1) discretization of the surface into rectangular regions
(2) setting up of transformation matrix such that the cutting plane becomes the z=0 plane
(3) transformation of surface data points using transformation obtained in (2)
(4) identification of dicretized regions through which the cutting plane passes
(5) calculation of intersection points in each of the regions identified in (4)
(6) setting up of inverse of transformation obtained in (2)
(7) drawing of cutting plane and intersection curve.

2.8.1 Discretization of the surface.

The surface points are taken in order of increasing u and v and are discretized into rectangular regions (not necessarily planar) in the u-v plane. The number of regions is a function of the complexity of the surface. It is assumed that the surface does not contain any interpenetrating areas.

This is a preliminary step both for intersections as well as for hidden surface display.

2.8.2 Transformation.

To ease up the calculations required for intersection calculations a transformation of the coordinate system to the cutting plane is required. This consists of translating the object coordinate system to a point on the cutting plane and then aligning the z-axis with the normal to the cutting plane. This results in searching for those regions of the surface where z=0 lies. The steps taken in achieving this transformation is given below ;

Let the three non-collinear points defining the cutting plane be p_1, p_2 and p_3 The normal to the plane from the origin of the coordinate system is then given by

$$\underline{n} = \frac{(\underline{p}_3 - \underline{p}_1) \times (\underline{p}_2 - \underline{p}_1)}{|(\underline{p}_3 - \underline{p}_1) \times (\underline{p}_2 - \underline{p}_1)|} \qquad 2.8.2.1$$

the distance of this plane from the origin is given by

$$d = \underline{n} \cdot \underline{p}_1 \qquad 2.8.2.2$$

the point at which the normal intersects the plane is

$$\underline{r} = d\,\underline{n} \qquad 2.8.2.3$$

Setting up a translation transformation so that the origin of the coordinate system gets mapped on to the point defined by the vector $\underline{r}$, the translation matrix is

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -r_x & -r_y & -r_z & 1 \end{bmatrix} \qquad 2.8.2.4$$

Since the normal to the cutting plane is to be aligned to the z-axis of the coordinate system, the z-axis direction is defined by $\underline{n}$. Taking an arbitrary vector lying on the cutting plane, say $\underline{b} = (\underline{p}_1 - \underline{r})$ and assigning this to the y-axis direction, and the vector product $\underline{a}$ of $\underline{b}$ and the normal vector, as the x-axis direction, a rotation matrix R can be set up

$$R = \begin{bmatrix} a_x & b_x & n_x & 0 \\ a_y & b_y & n_y & 0 \\ a_z & b_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 2.8.2.5$$

Postmultiplying equation (2.8.2.4) with (2.8.2.5) results in the required transformation matrix T

$$T = \begin{bmatrix} a_x & b_x & n_x & 0 \\ a_y & b_y & n_y & 0 \\ a_z & b_z & n_z & 0 \\ -r_x & -r_y & -r_z & 1 \end{bmatrix} \qquad 2.8.2.6$$

The inverse of this matrix is also required for later back-transformation to the original coordinate system.

2.8.3 Calculation of intersection points.

Setting up a parametric interpolation function, let the z-values of a rectangular region of a surface saddle a zero, with f_{1z} and f_{2z} the z-values at the two vertices between which there is a intersection point, then

$$(1-t).f_{1z} + t.f_{2z} = 0 \qquad 2.8.3.1$$

where t is the parametric variable lying in the interval

$$0 \leq t < 1 \qquad 2.8.3.2$$

therefore

$$t = \frac{f_{1z}}{f_{2z}-f_{1z}} \qquad 2.8.3.3$$

Then using the value of t obtained in the above equation the x and y-values of the intersection points are given by

$$x = (1-t).f_{1x} + t.f_{2x} \quad , \text{ and}$$

$$y = (1-t).f_{1y} + t.f_{2y} \qquad 2.8.3.4$$

2.9 Hidden Surface Display.

The rectangular vertices obtained after discretization of the surface are in the object coordinate sytem, and therefore they have to be transformed to the viewing plane coordinates before any hidden surface algorithm can be applied.

The surface data points are first transformed to the viewing plane. It can be assumed that the rectangular regions are almost planar and that they do not interpenetrate. The average depth of each rectangle is therefore the mean of the z-values of the four vertices.

Using a bubble sort the rectangles are arranged in depth-order with the rectangle farthest from the viewer appearing first in the list of sorted rectangles.

The sorted list of rectangles is then accessed and and each is filled with a colour selected by the user. The rectangle drawn last covers up the already drawn rectangles, thereby creating a hidden surface view. With this technique the hidden line problem is trivially solved if the fill-colour used is the background colour.

CHAPTER III

SURF-3D

3.1 Overview of the system.

SURF-3D is a graphic interactive, menu-driven software for the design and display of three-dimensional surfaces. The formulations described in sections 2.2, 2.3 and 2.4 have been implemented in this package, thereby facilitating the design of spline-blended, Bezier and B-spline surfaces. A comprehensive set of viewing parameters have been introduced to enable the user to visualize the surface in any orientation, either in axonometric or perspective projections. In addition hidden surface and hidden line views can also be obtained to gain more realism.

The surfaces can be modified by changing the positions of control points which may be graphically 'picked' from the screen for input to the program.

Facilities for general curve generation for later use in designing swept, axi-symmetric surfaces have also been included as a special feature of the software.

Input data may either be created on-line or may be given as input from a file, thereby making it easy for available surface data to be input.

All data regarding the control points, curve data

points, and surface data points can be obtained as output from the program in a form suitable for documentation purposes.

The entire system is modular in structure and open-ended to the extent that additional features may be added in future without having to re-edit or change the package. All the modules have been written using FORTRAN-77 and are implemented in the Norsk Data 560/CX computer system. The program presently runs on Tektronix 4107 and 4109 graphic terminals. Changes may be required if the package is to be run on other systems. The underlying basic graphic action routines are built-up on the PLOT-10 Terminal Control System software, a product of Tektronix Inc. An entire suite of three-dimensional wireframe graphics routines has been implemented in this package.

The data structures used are simple and are close to their actual mathematical equivalents e.g. a set of vectors has been represented as a three-dimensional array in the program.

3.2 Interactive menu features.

The system provides an interactive menu, which helps in going through the steps of input, design and display in a systematic manner. The user at any stage is given a display of all the commands available to him at that stage. The

main advantage of this is that the user does not have to commit to memory any of the commands.

There are in all six menus viz.

(i) MAIN

(ii) VIEW

(iii) CURVE

(iv) FIT CURVE

(v) SURFACES

(vi) SWITCHES

Each of these above provides sets of options or commands which can be selected when the program displays a particular menu. The user, for example, may input data from a file using the MAIN, then set the viewing parameters using VIEW and then on to surface generation and display using SURFACES, which gives a choice of selecting among the three different types of surfaces presently supported, or for intersections and hidden surface displays.

The selection process has been made simple. The user needs only to align the cursor with a particular option and then press the space-bar to complete the selection. The program gives an immediate feedback if a proper selection is made.

The overall structure of the menu system is modular and new commands and options may be added as and when required. The structure of the menu system has been depicted in figure (3.2.1).

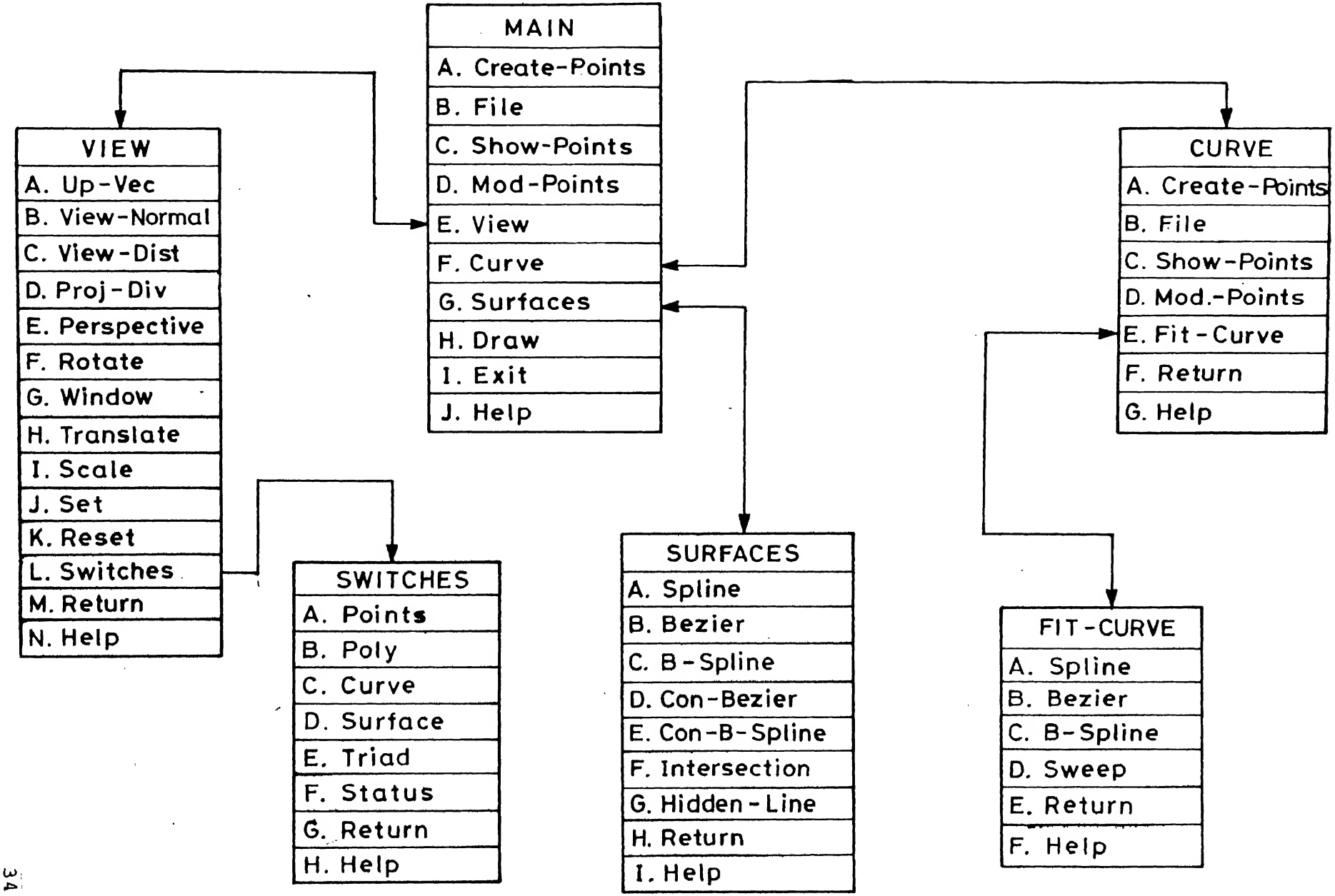


Fig. 3.2.1 Menu Structure.

3.3 Graphics features.

A full-fledged three dimensional wire-frame graphics system has been implemented as a part of the package. This has been built up as a super-set over the set of basic graphics action routines provided by the PLOT10- Terminal Control System. This therefore, has resulted in efficient use of all the graphic facilities offered by Tektronix 4100 series colour graphic terminals.

The following are the features of the three-dimensional graphics system :

(i) full three-dimensional windowing has been provided so that user-specified coordinate values can be easily mapped to the screen-coordinate system. The user works in an environment in which he neither requires knowledge of computer graphics nor of computer programmning.

(ii) three dimensional viewing transformations encompassing both parallel and perspective projections, including independent translation, scaling and rotation about any axis are possible

(iii) hidden surface display using panel-filling of surface patches with user selectable colours.

3.3.1 Display options.

The three-dimensional graphics routines support full colour viewing transformations in the principle of a synthetic camera. The user can change the view parameters like direction of projection, center of projection or eye-point, view-up direction, view distance, view plane position and orientation etc. to arrive at a required view for display.

In addition, a user may specify independent three-dimensional coordinate axes rotations, translations, and scaling transformations for viewing.

Facilities have been provided to selectively display entities like points, polyhedrons or polygons, curves, surfaces, axes triad etc. This allows the user to specify explicitly what he wants to display. Points are displayed as identifiable marked points, with control polygons and polyhedrons appearing in straight lines. The surfaces and curves are drawn as families of curves.

3.3.2 Viewing transformation.

The view parameters that the user can set within the program are :

(i) the view reference point (XR,YR,ZR); the rotation, translation and scaling transformations are all done about this point. (The default value is (0,0,0))

(ii) the view plane normal (VPX,VPY,VPZ) is a vector which is perpendicular to the view plane and pointed towards the view reference point. (The default is the z-axis (0,0,-1))

(iii) the view distance (VD) is the distance of the view plane from the view reference point(default 0)

(iv) the view-up direction vector (VPX,VPY,VPZ); the direction which should be vertical, when the viewing transformation is applied(default being the y-axis (0,1,0))

These parameters allow the user to select a particular view of the surface.

For parallel projection, the user must specify the direction of projection (PX,PY,PZ)(default (0,0,1)), whereas in the case of perspective projections, the user has to specify the center of projection point or eye position (XC,YC,ZC).

Once all of these have been set by the user, the following steps outline the transformation procedure :

Step 1. Translation of the object coordinate system to the view plane origin;

Step 2. Rotation about the x-axis to place the view plane normal in the x-z plane;

Step3. Rotation about y-axis to align the z-axis of the

coordinate system with the view plane normal;

Step 4. Rotation about z-axis to align the view-up vector with the y-axis of the coordinate system;

Step 5. If perspective projection is specified, then transformation of the eye point to the view plane coordinate system;

else

if parallel projection, then transformation of the parallel projection vector to the new coordinate system.

3.4 Surface modules

The following surface modules have been implemented:

(i) Spline surface

(ii) Bezier surface

(iii) B-spline surface

3.4.1 Spline surface.

Input to this module consistes of

(i) the rectangular array of control points

(ii) number of intermediate points per span

Assuming free ends for each curve on the surface unless otherwise specified, a general spline curve segment between points $\underline{r}_i$ and $\underline{r}_{i+1}$ and using equation (2.2.3)

$$\underline{r}_i(w) = \underline{r}_i.(1-3w^2+2w^3) + \underline{r}_{i+1}.(3w^2-2w^3) + \underline{r}'_i.(w-2w^2+w^3) \quad 3.4.1$$
$$+ \underline{r}'_{i+1}(-w^2+w^3).$$

Differentiating with respect to w

$$\underline{r}'_i(w) = \underline{r}_i(-6w+6w^2) + \underline{r}_{i+1}(6w-6w^2) + \underline{r}'_i(1-4w+3w^2) + \underline{r}'_{i+1}(-2w+3w^2) \qquad 3.4.2$$

and again

$$\underline{r}''_i(w) = \underline{r}_i(-6+12w) + \underline{r}_{i+1}(6-12w) + \underline{r}'_i(-4+6w) + \underline{r}'_{i+1}(-2+6w) \qquad 3.4.3$$

the slopes $\underline{r}'_i$s are unknown; substituting w=0 and w=1 in the equation (3.4.3)

$$\underline{r}''_i(0) = -6\underline{r}_i + 6\underline{r}_{i+1} - 4\underline{r}'_i - 2\underline{r}'_{i+1} \qquad 3.4.4$$

$$\underline{r}''_i(1) = 6\underline{r}_i - 6\underline{r}_{i+1} + 2\underline{r}'_i + 4\underline{r}'_{i+1} \qquad 3.4.5$$

Since second derivative continuity is assumed

$$\underline{r}''_i(1) = \underline{r}''_{i+1}(0) \qquad 3.4.6$$

therefore from equations (3.4.4) and (3.4.5)

$$\underline{r}'_i + 4\underline{r}'_{i+1} + \underline{r}'_{i+2} = -3\underline{r}_i + 3\underline{r}_{i+2} \qquad 3.4.7$$

for free ends setting i=1

$$2\underline{r}'_1 + \underline{r}'_2 = -3\underline{r}_1 + 3\underline{r}_2 \qquad 3.4.8$$

similarly for the other end of the spline setting i=n-1

$$\underline{r}'_{n-1} + 2\underline{r}'_n = -3\underline{r}_{n-1} + 3\underline{r}_n \qquad 3.4.9$$

Using equations (3.4.8) and (3.4.9) for the ends and equation (3.4.7) for the other points of the spline

$$[A].[\underline{r}'] = [B].[\underline{r}] \qquad 3.4.10$$

where [A] is n x n matrix and [B] is a matrix with n rows Therefore

$$[\underline{r}'] = [A]^{-1}.[B].[\underline{r}] \qquad 3.4.11$$

which gives the required slopes in one direction at each of the points on the curve.

Substitution of the points and slopes in the equation (3.4.1) and stepping through unit intervals of the parametric variable w for each span, a curve on the surface is generated.

3.4.2 Bezier surface.

Input to this module consists of :

(i) the rectangular array of control points

(ii) number of sections on the surface(nsec)

The maximum size of the rectangular array of control points allowed by the program is 20 x 20. The number of sections (nsec) determines the number of surface points that are to be calculated. The number of surface points is given by (nsec+1) x (nsec+1). The maximum number of surface points that can be calculated by this module is 50 x 50.

The following steps outline the algorithm for generating a Bezier surface.

Step 1. Set number of points on each curve (np) = nsec+1

Step 2. Initialize u and v parametric values required for generating np x np points on the surface:

```
ustep = 1/(np-1)
vstep = 1/(np-1)
for i=1 to np do
     u[i] = (i-1) * ustep
     v[i] = (i-1) * vstep
od;
```

Step 3. Repeat steps 4 to 6 for each value of u.

Step 4. Find blending function values using equations (2.3.1) and (2.3.2)

Step 5. Calculate intermediate curve-defining points

using equation (2.3.3)

Step 6. Repeat steps 7 to 8 for each value of v

Step 7. Find blending function values in v-direction using equations (2.3.1) and (2.3.2)

Step 8. Calculate surface points using equation (2.3.4).

Step 9. Draw surface.

3.4.3 B-spline surface.

The input to this module consists of

(i) the rectangular array of control points

(ii) number of sections on surface(nsec)

The maximum size of the rectangular array of control points allowed by the program is 20 x 20. The number of sections (nsec) determines the number of surface points that are to be calculated. The number of surface points is given by (nsec+1) x (nsec+1). The maximum number of surface points that can be calculated by this module is 50 x 50.

The following steps outline the algorithm for generating a B-spline surface.

Step 1. Set number of points on each curve (np) = nsec+1

Step 2. Calculate the knot values in both u and v directions using equations 2.4.4 and 2.4.5

Step 2. Initialize u and v parametric values required for generating np x np points on the surface:

$ustep = (t_{m+k})_u / (np-1)$

$vstep = (t_{n+k})_v / (np-1)$

```
for i=1 to np do
        u[i] = (i-1) * ustep
        v[i] = (i-1) * vstep
od;
```

Step 3. Repeat steps 4 to 6 for each value of u.

Step 4. Find blending function values using equations (2.4.1) and (2.4.2)

Step 5. Calculate intermediate curve-defining points using equation (2.4.3)

Step 6. Repeat steps 7 to 8 for each value of v

Step 7. Find blending function values in v-direction using equations (2.4.1) and (2.4.2)

Step 8. Calculate surface points using equation (2.4.7).

Step 9. Draw surface.

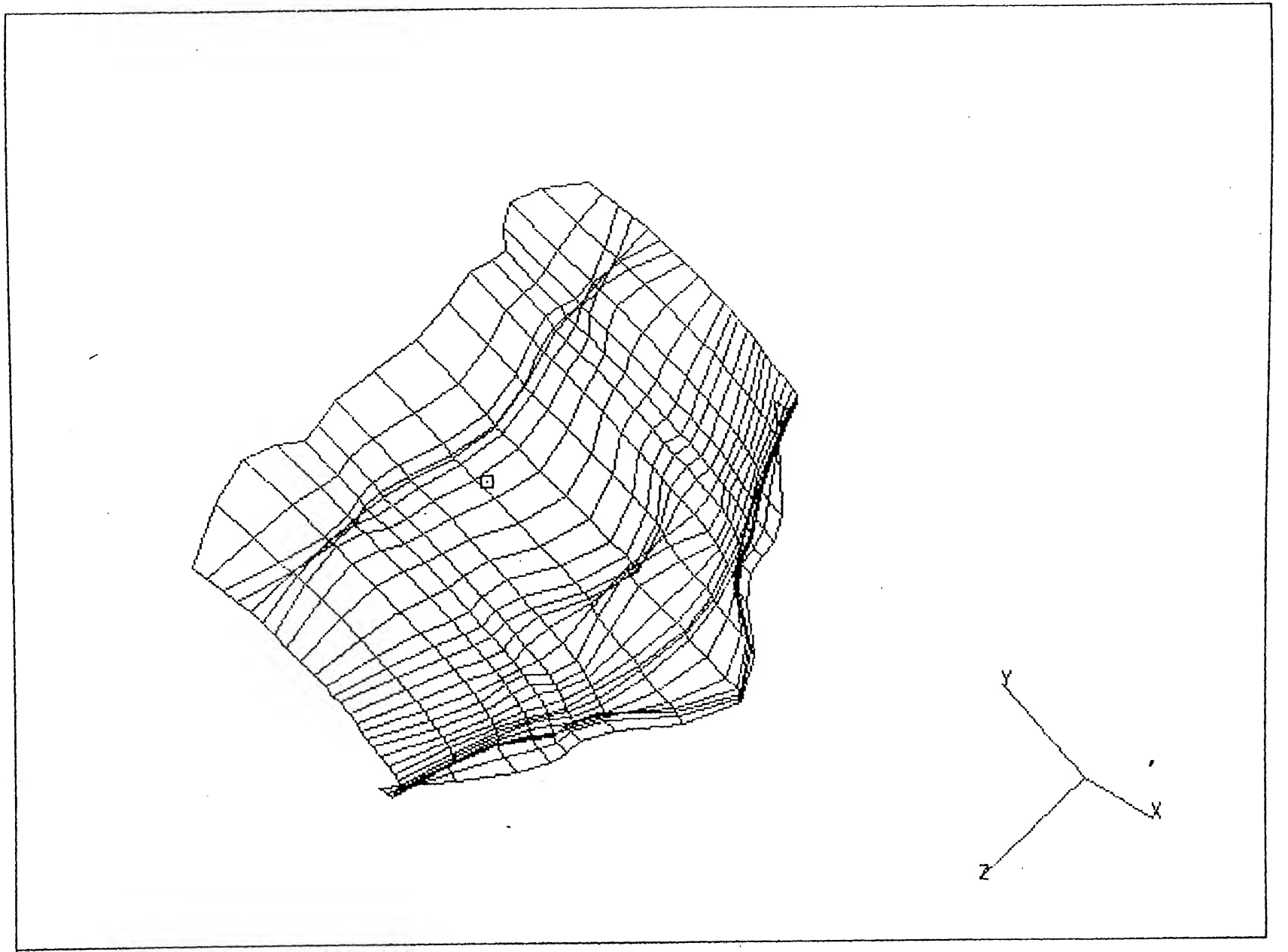
CHAPTER IV

CASE STUDIES

4.1 Examples of spline blended surfaces.

The surface passes through all the given control points in a spline-blended surface. Therefore a change in the position of a control point will lead to change in the entire surface because second derivative continuity has to be maintained throughout any curve on the surface. The alterations that occur in a spline blended surface are usually unpredictable and therefore not suitable for modifications. It is advisable only to input available surface data using this representation and do all modifications after a conversion either to a Bezier or a b_spline surface. See figure 4.1, which is a graphic output of the design package.

Figure 4.1. Program output of a spline-blended surface

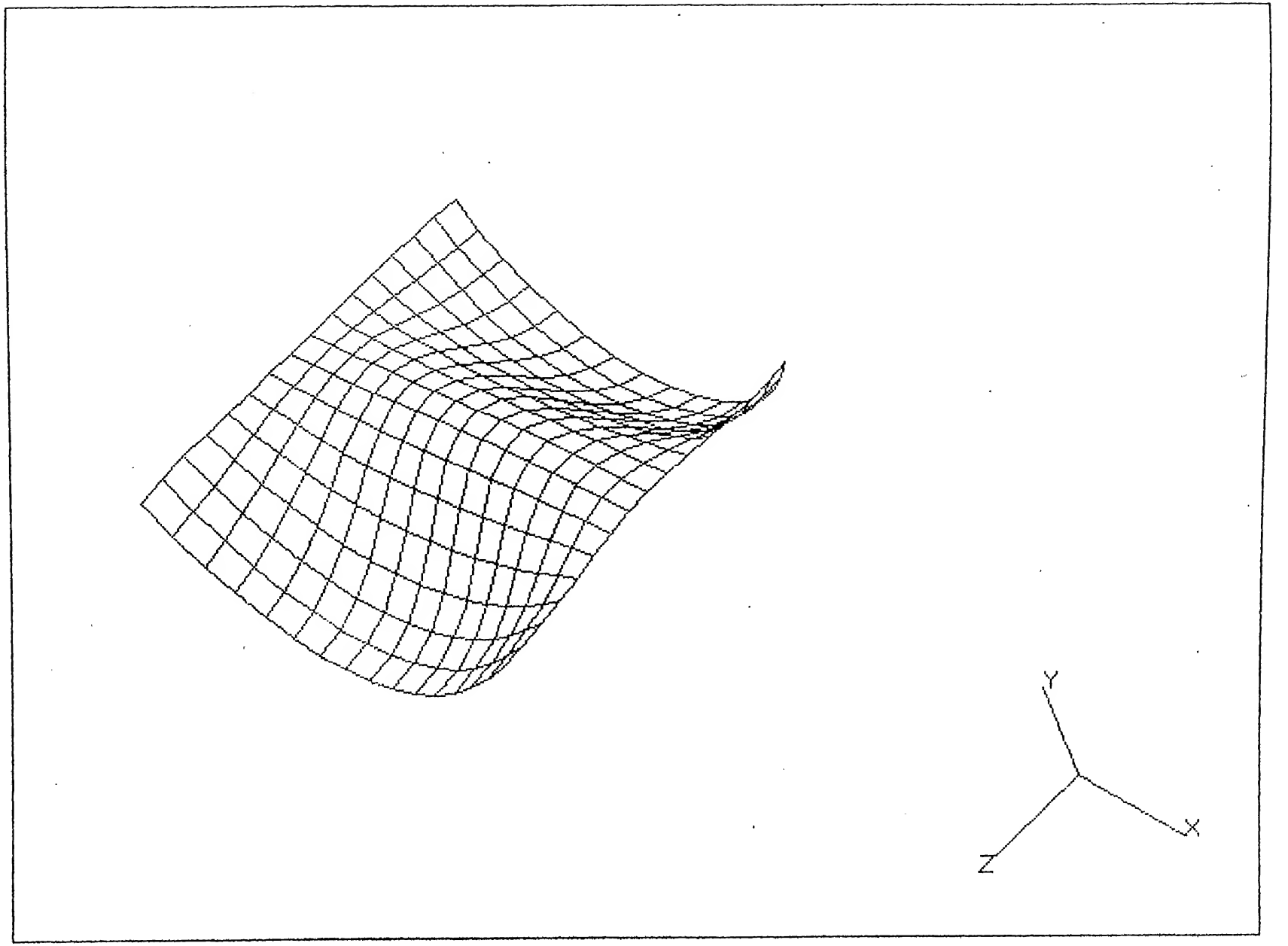


4.2 Examples of Bezier surfaces.

Bezier surfaces only approximate the control points, as stated before. Only the four corner control points lie on the surface. The blending functions used for Bezier surfaces of different orders give it the convex-hull property. The surface is contained entirely within the polyhedron of control points. Modifications to positions of control points result in the surface getting altered more near the control point and in varying degrees over the rest of the surface. However, the changes are quite predictable, and with a little practice a user can achieve the design he requires.

The main drawback of a Bezier surface is that the order of the surface is explicitly controlled by the number of control points, and therefore the degree of the resulting surface is usually unalterable. Another shortcoming of this scheme is the global nature of change that occurs when a control point is modified. This does not allow a designer to make fine adjustments to the surface. See figure 4.2.

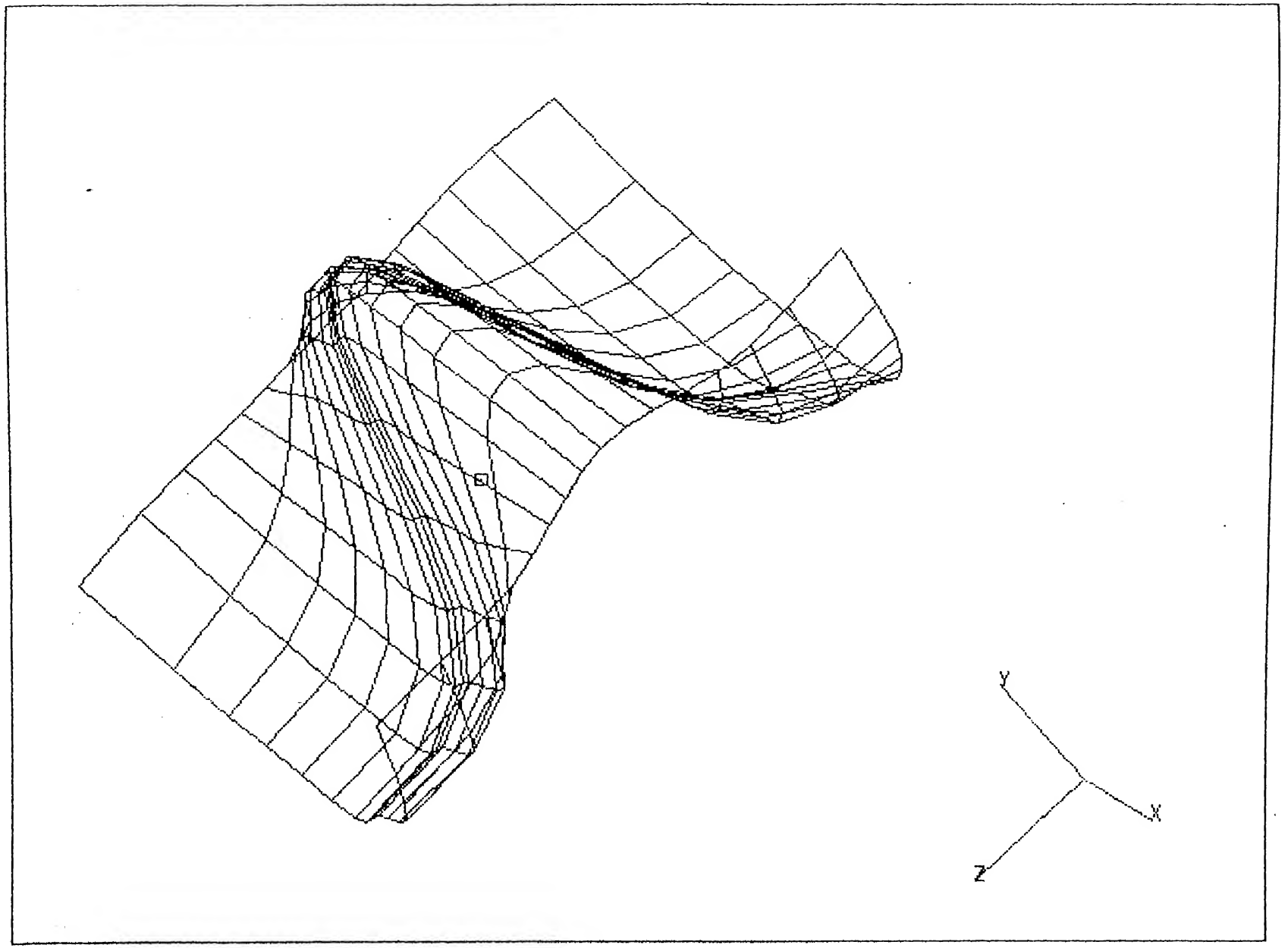
Figure 4.2. Program output of a Bezier surface



4.3 Examples of B-spline surfaces.

A b-spline surface behaves in the same way as a Bezier surface, only it follows the control polyhedron more closely. The level of continuity of the interpolated surface can be explicitly controlled by the designer and therefore its complexity. Also the modifications are of local nature. These two properties makes the b-spline representation one of the most popular schemes for surface design. See figure 4.3.

Figure 4.3. Program output of a B-spline surface



4.4 Examples of Intersections.

Figure 4.4.1 and 4.4.2 show intersection curves resulting out of a surfaces being cut by a plane. The intersection curve may be obtained for a surface oriented in any view.

Figure 4.4.1. Intersection

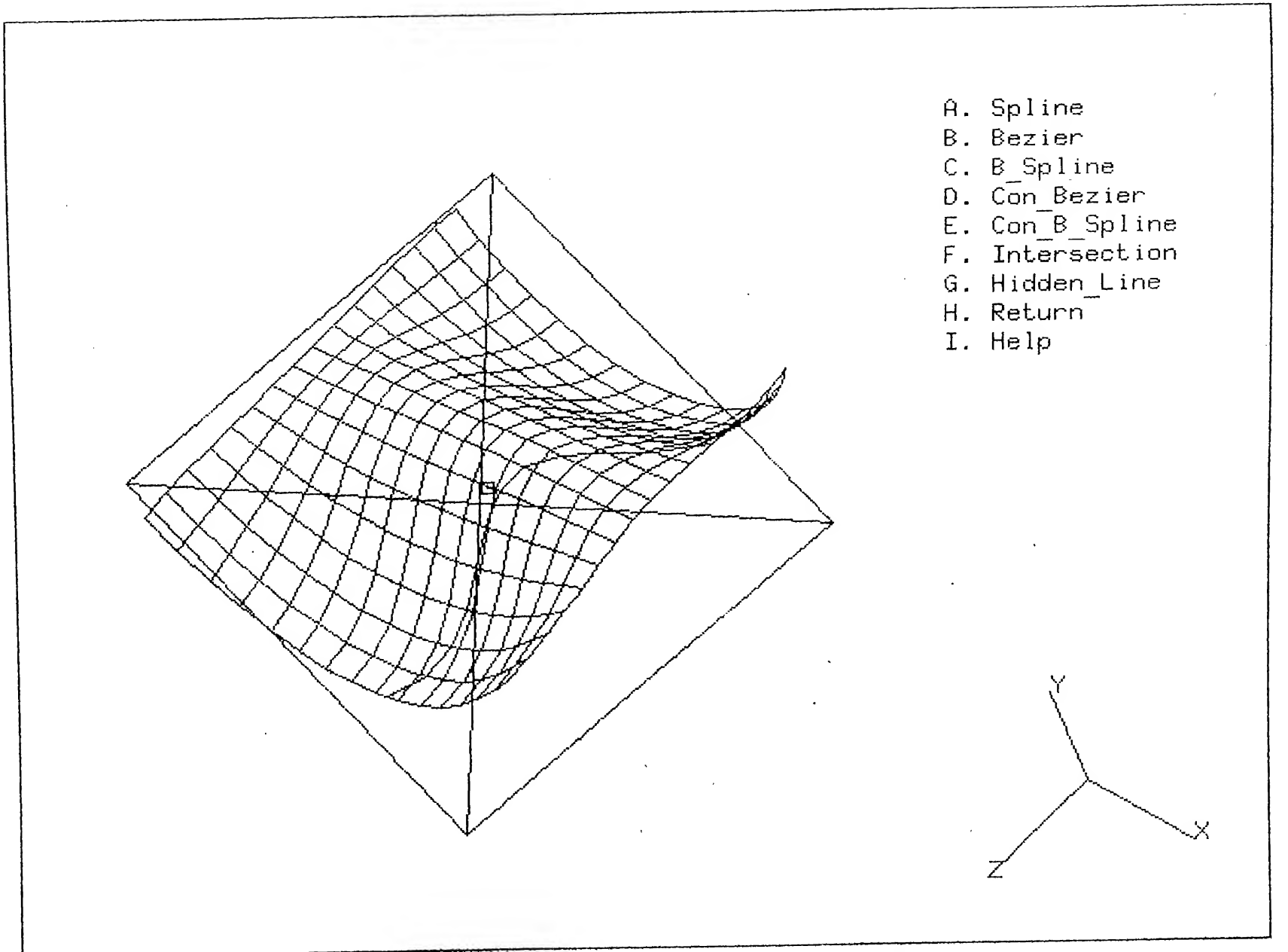
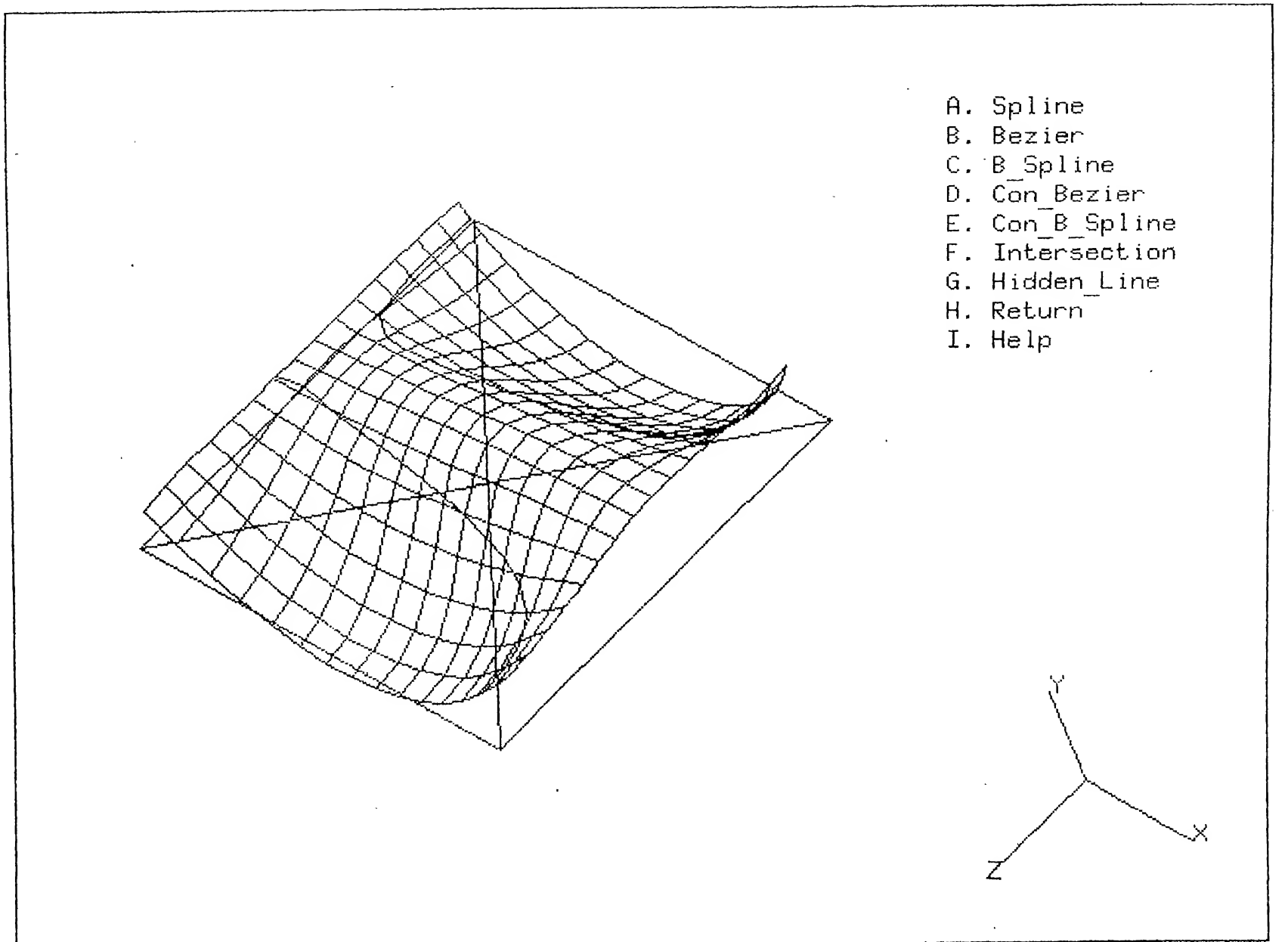


Figure 4.4.2. Intersection



4.5 Examples of editing surfaces.

Figure 4.5.1 shows a spline surface. The position of one of the control points was changed and a surface refitted. Note the alteration in the overall surface in figure 4.5.2.

Figure 4.5.3 is a Bezier surface, which when subjected to editing resulted in the surface in figure 4.5.4.

The B-spline surface depicted in figure 4.5.5 was edited in a similar fashion. Note the local change in the surface after editing in figure 4.5.6.

Figure 4.5.1. Spline surface before editing

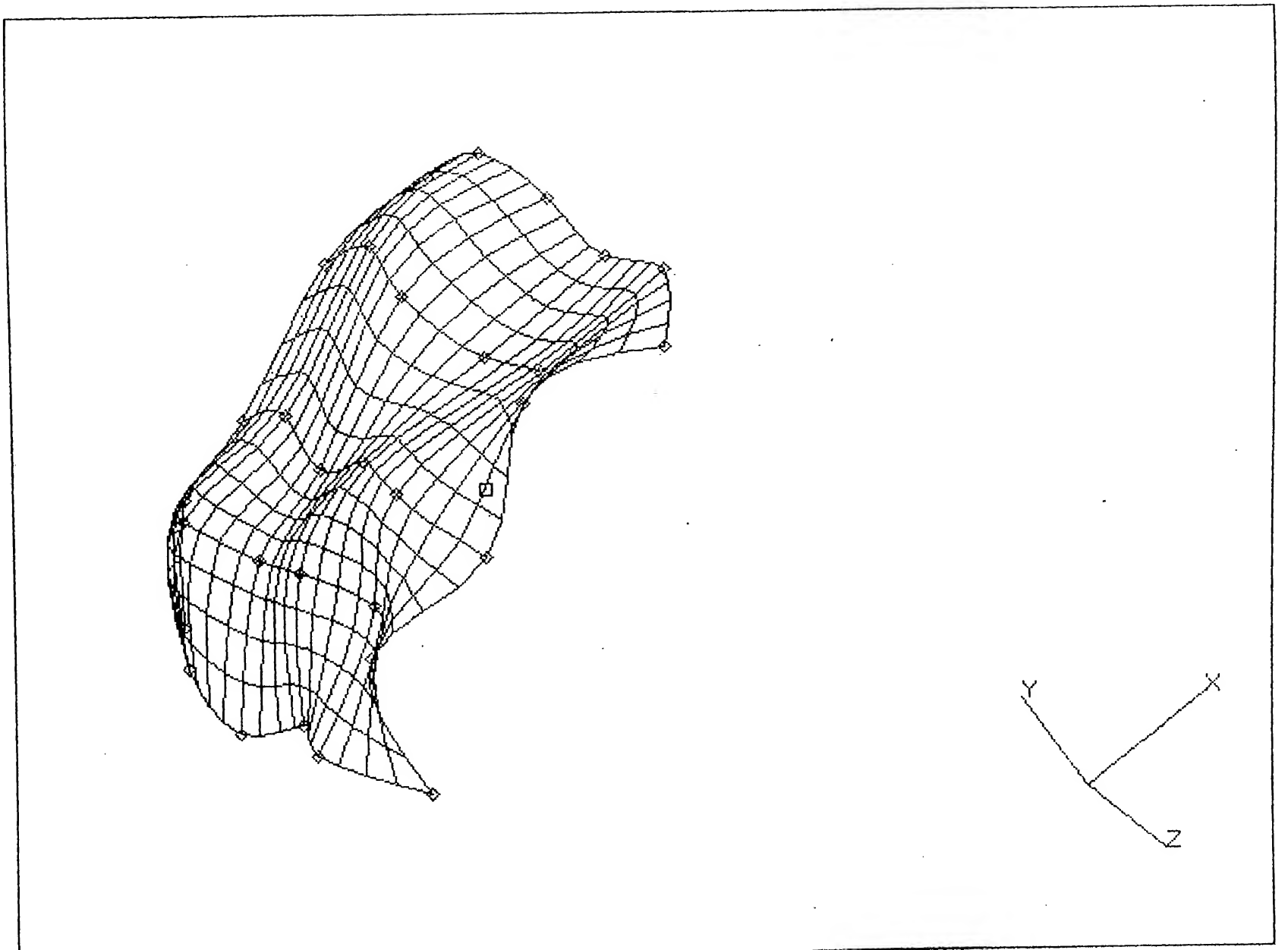


Figure 4.5.2. Spline surface after editing

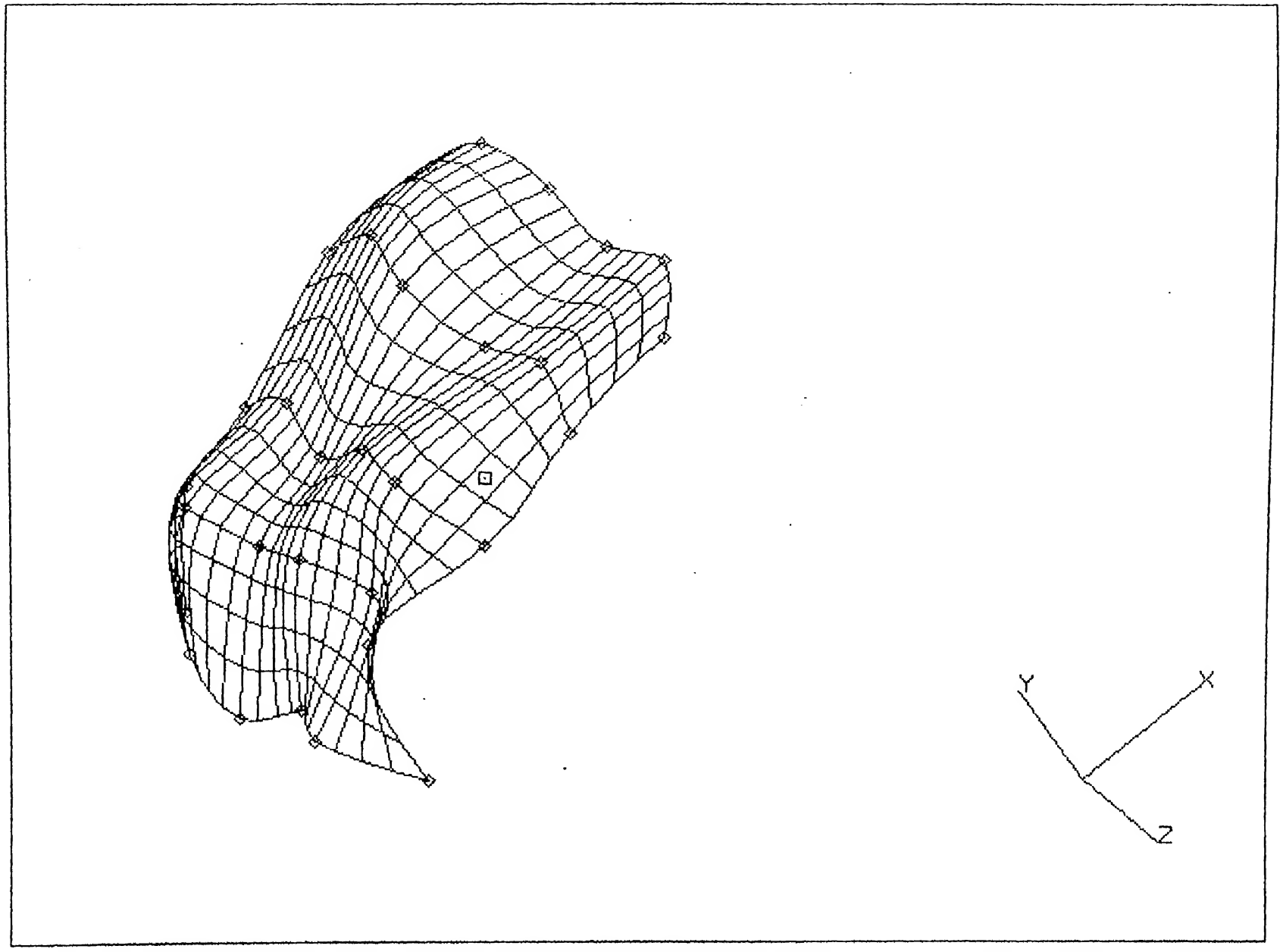


Figure 4.5.3. Bezier surface before editing

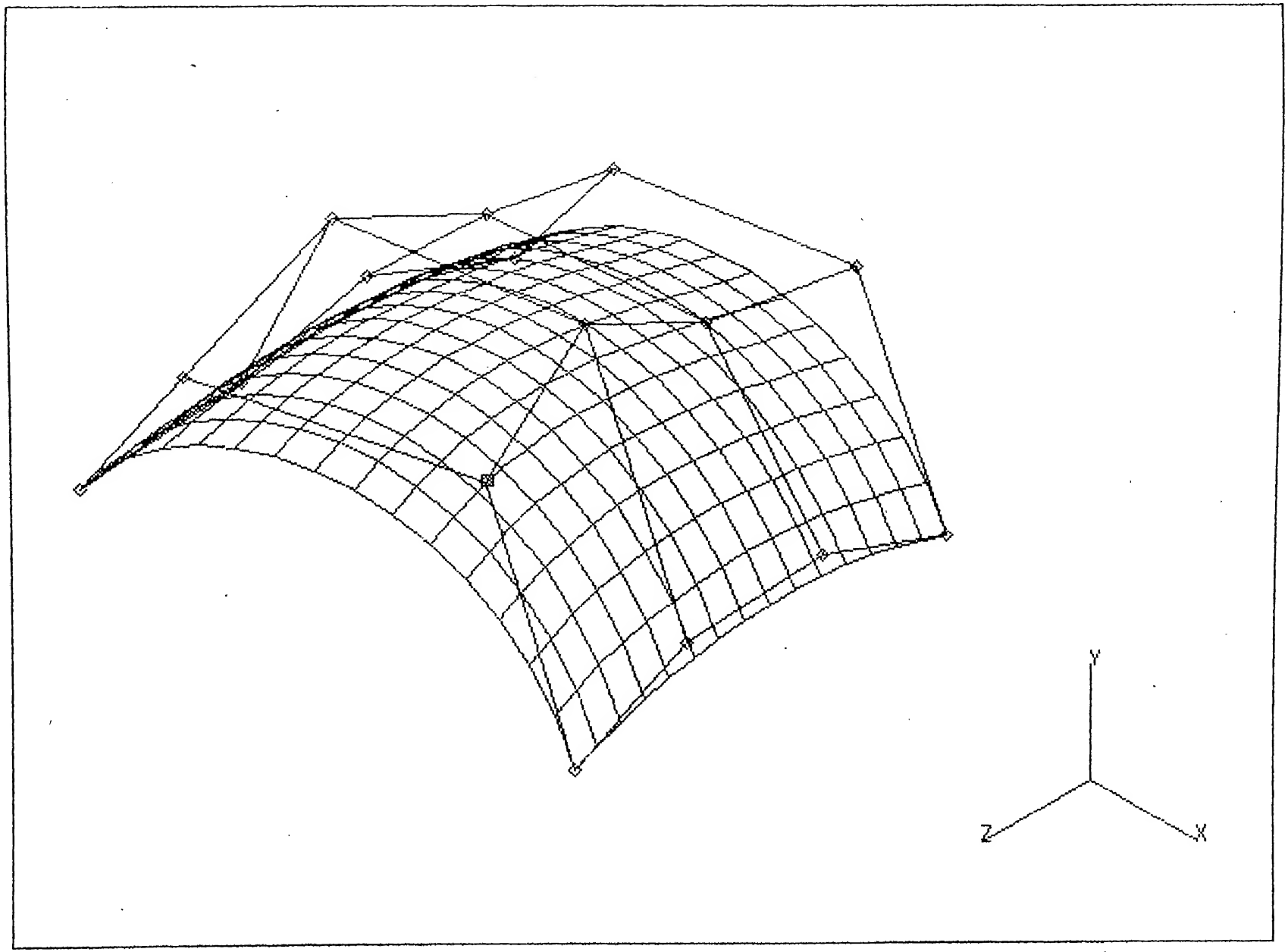


Figure 4.5.4. Bezier surface after editing

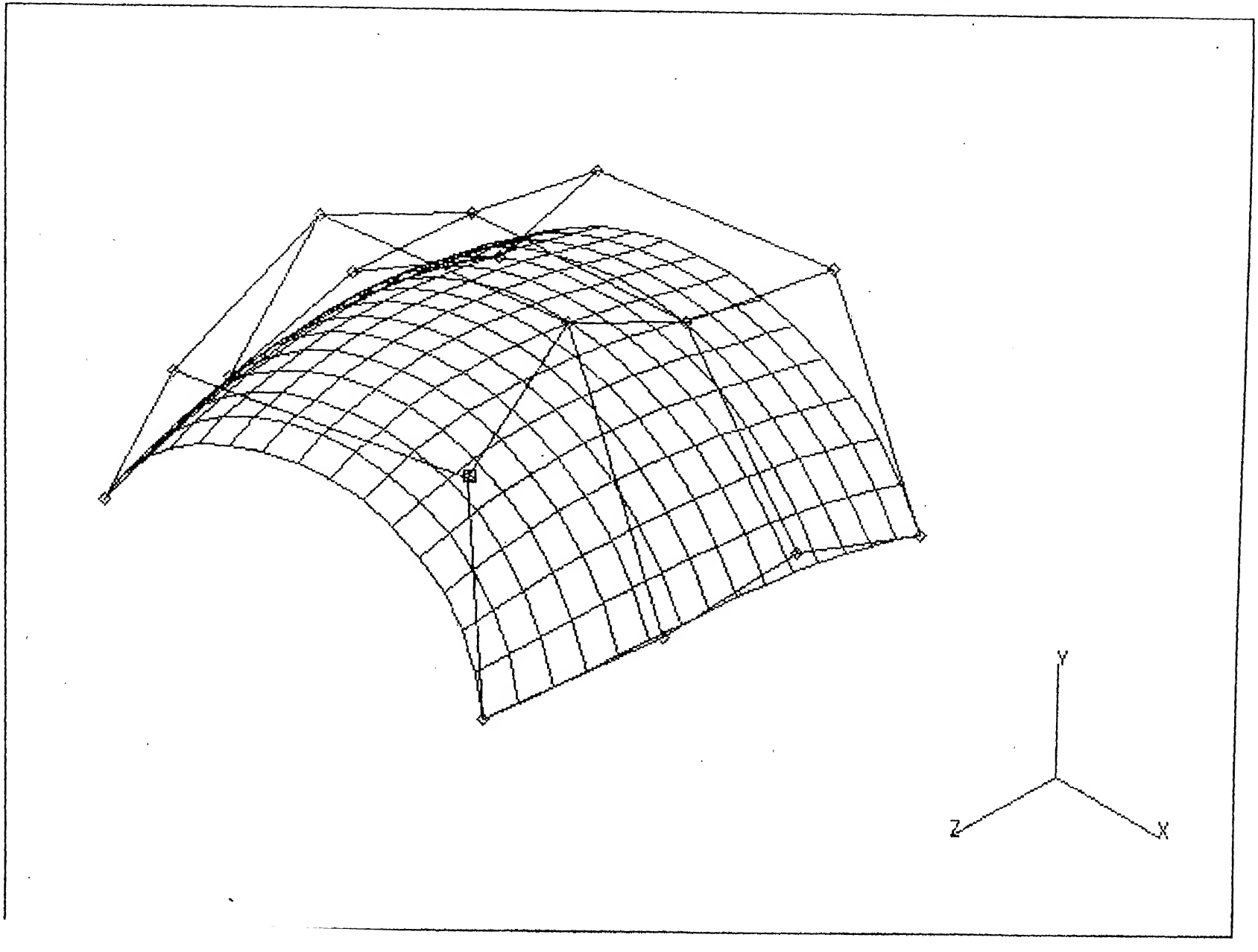


Figure 4.5.5. B-spline surface before editing

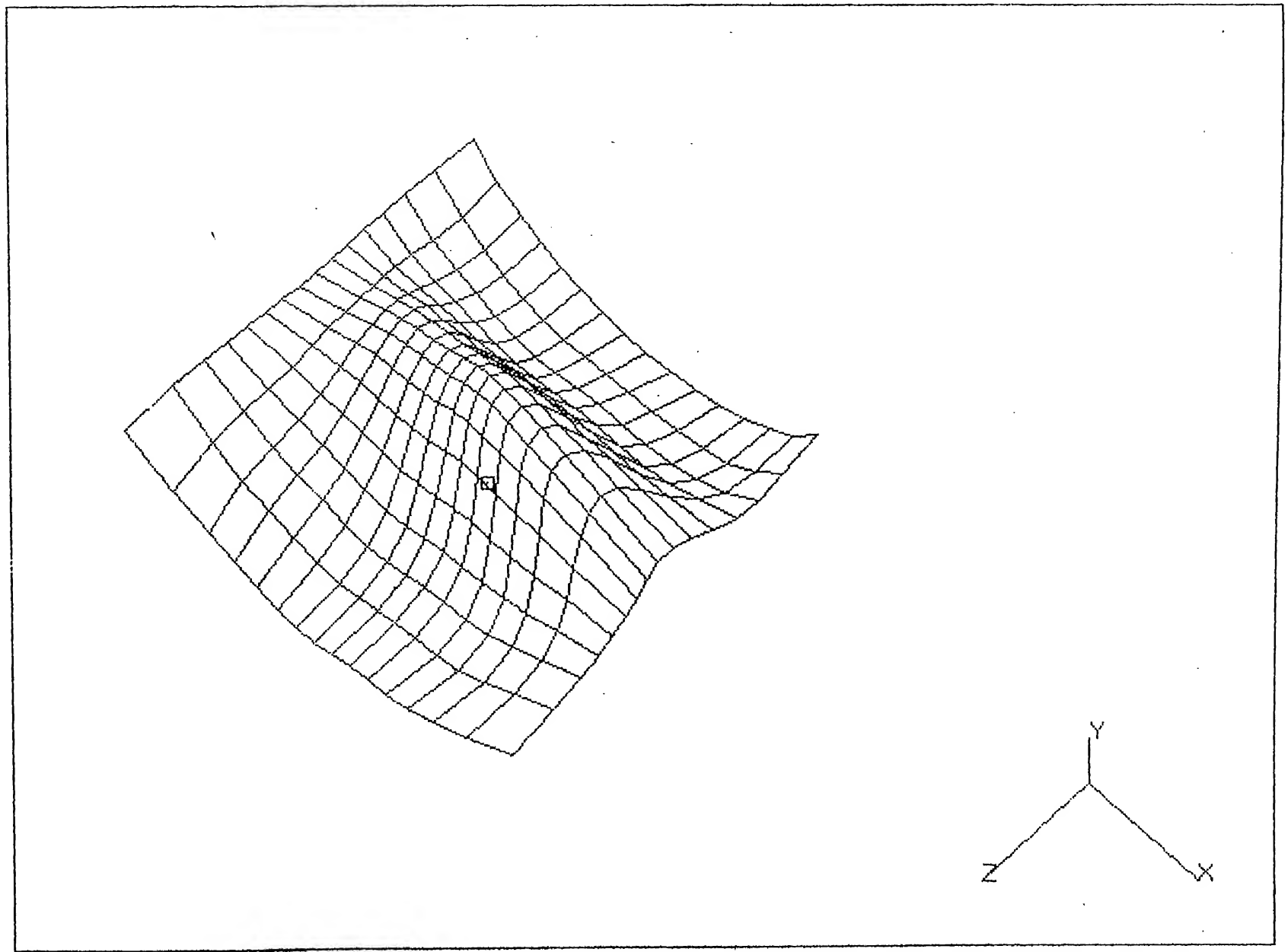
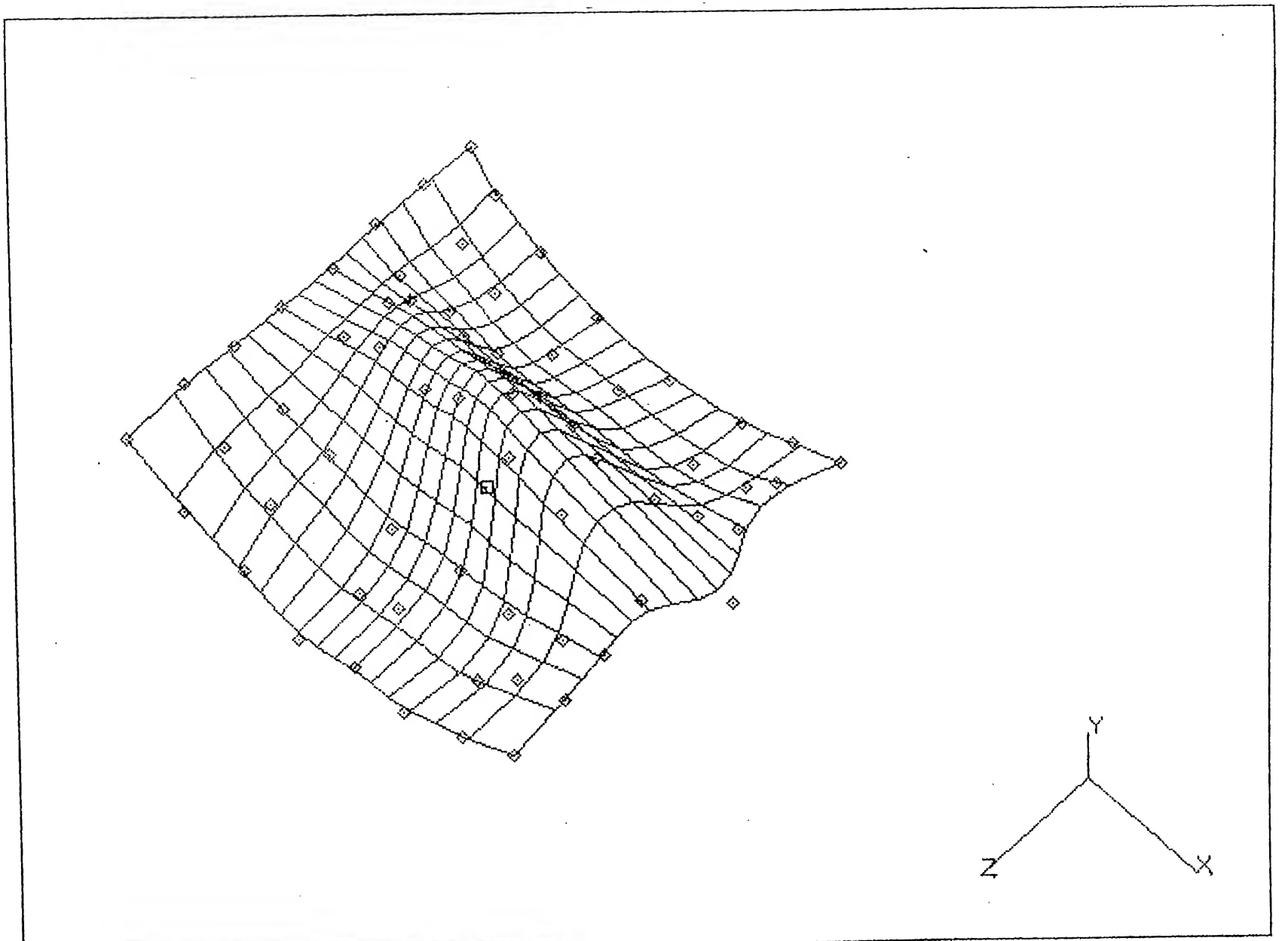


Figure 4.5.6. B-spline surface after editing



4.6 Examples of axi-symmetric surfaces.

Figure 4.6.1, 4.6.2 and 4.6.3 are illustrations of swept axi-symmetric surfaces, which can be designed by the package.

Figure 4.6.1. An axi-symmetric component (360 deg)

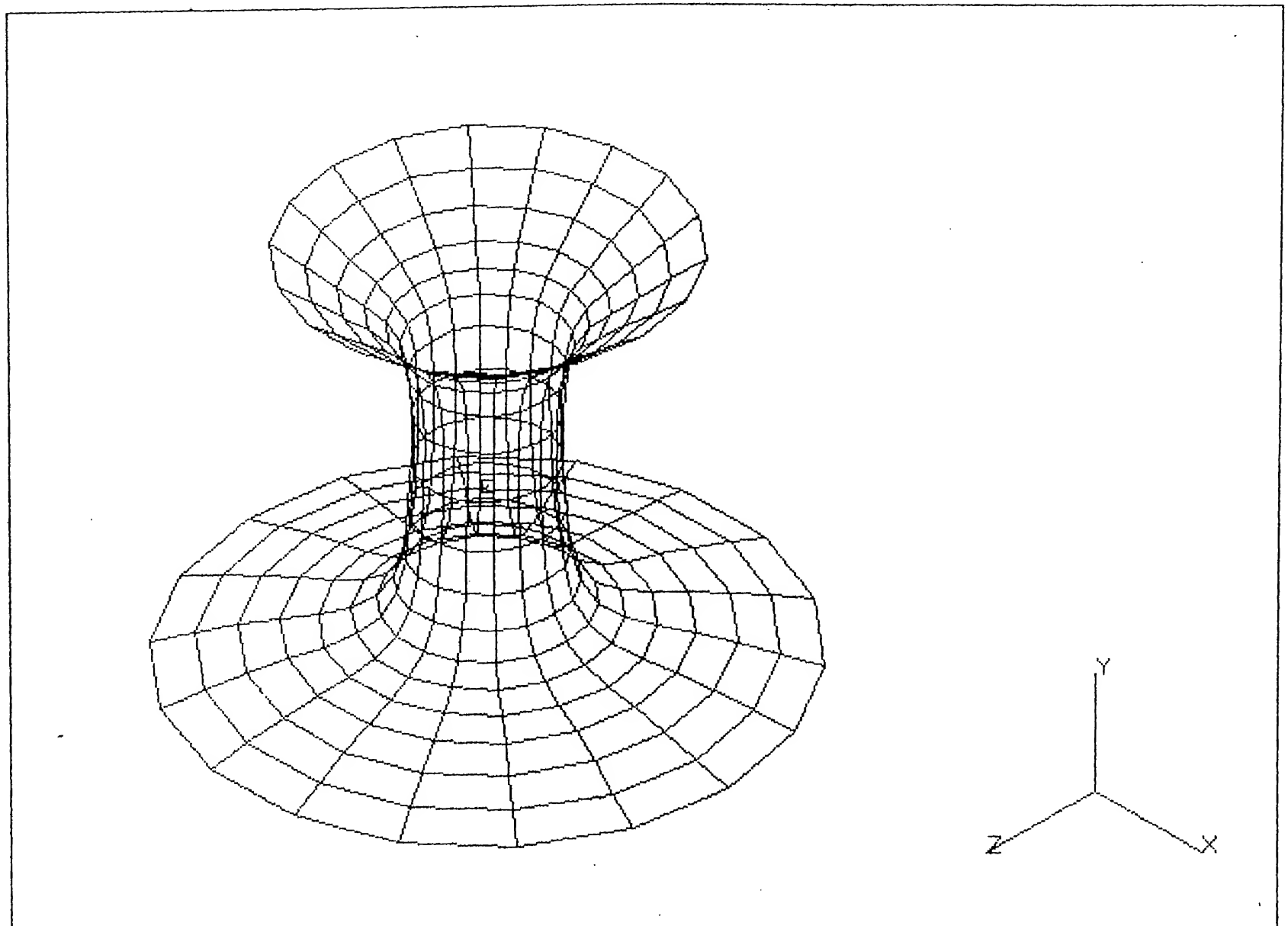
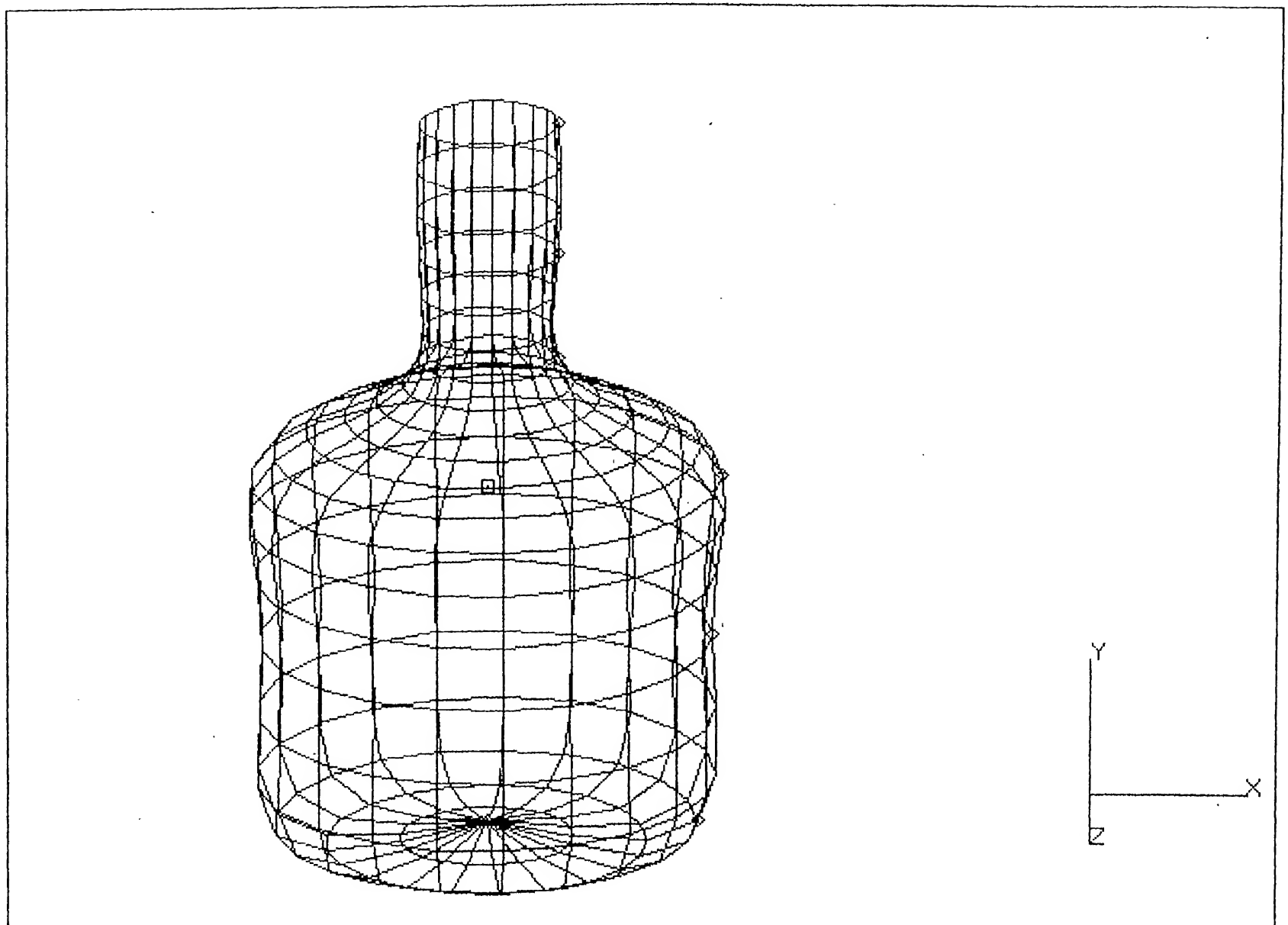


Figure 4.6.3. An axi-symmetric component (360 deg)



4.7 Examples of hidden surface removal.

Figure 4.7.1 is a program output of a surface with hidden faces removed by the algorithm described in chapter III. Figure 4.7.2 is another example. The same algorithm can be used to generate a hidden line view of a surface.

Figure 4.7.1. Hidden surface display

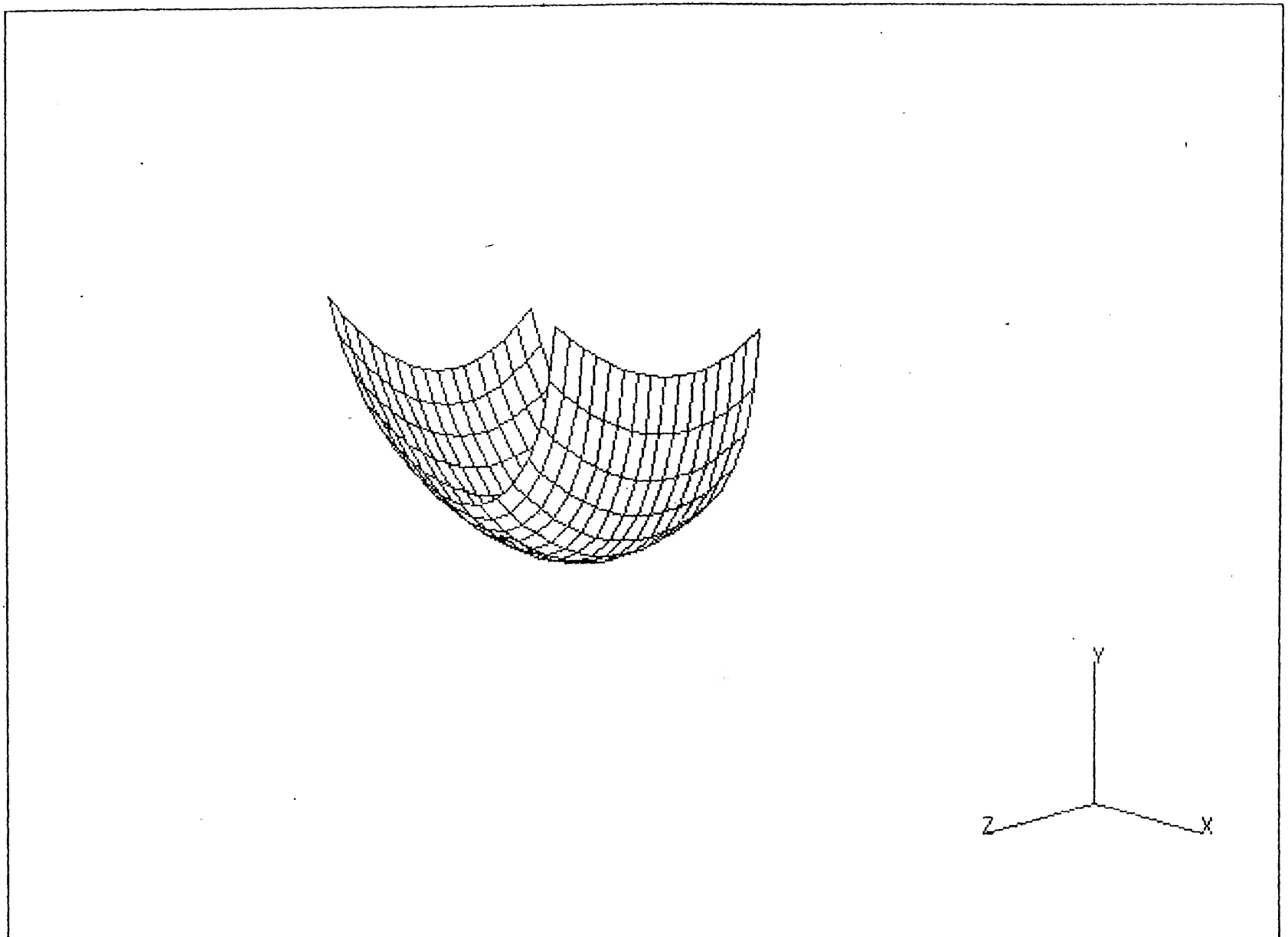
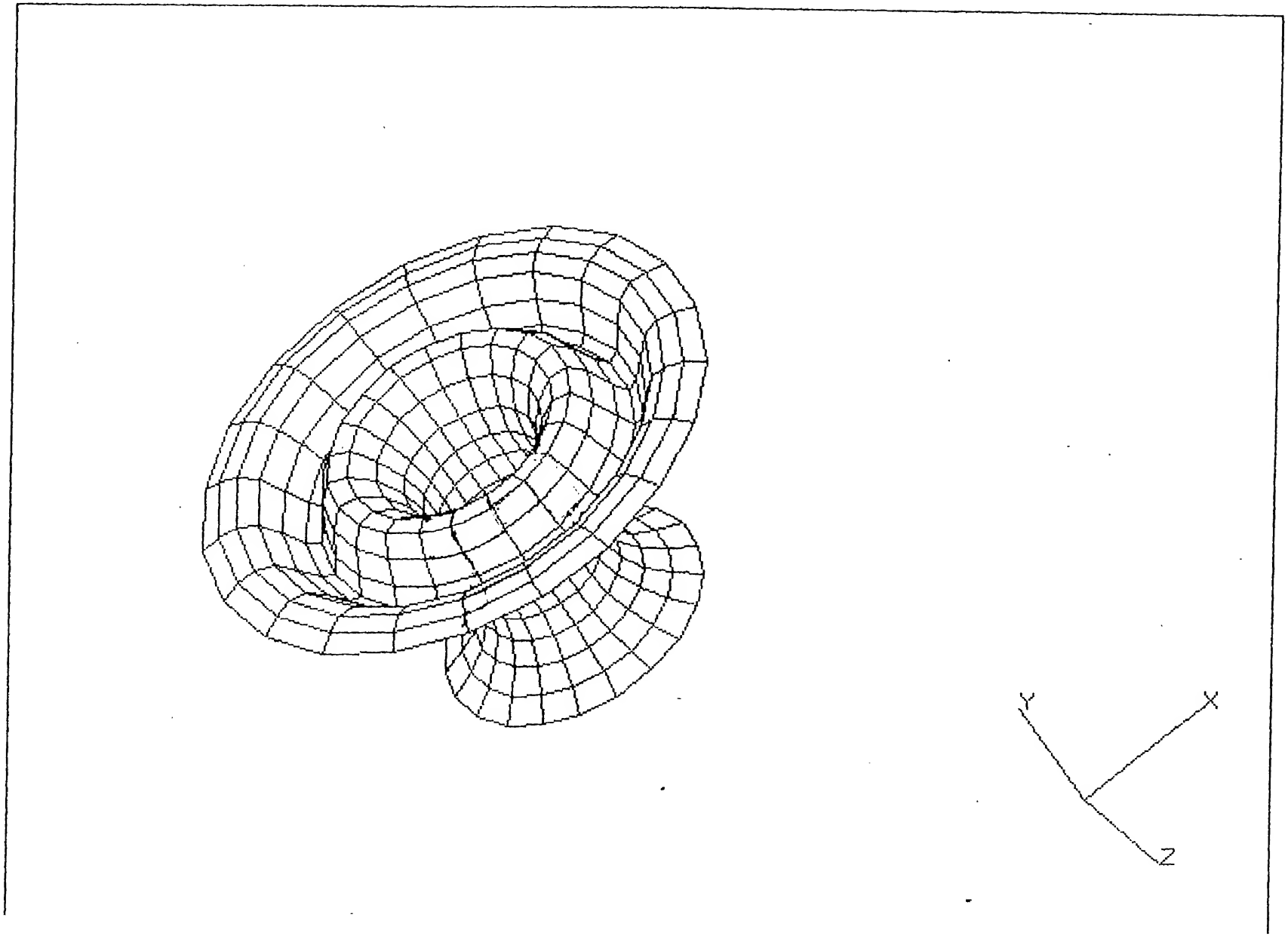


Figure 4.7.2 Hidden surface display



CHAPTER V

CONCLUSIONS

5.1 Technical Summary

Designers and draftsmen use the classical 'lofting' techniques to design surfaces. These methods are based on cross-sectional details obtained from templates. The entire procedure of design, keeping aesthetics and/or aerodynamics in view, is time consuming and is of a specialized nature. Proficiency in this craft takes years of experience in surface design and engineering and the methods have very limited applications in the field of free-form sculptured surfaces.

The emerging technology of computer graphics coupled with the advances made in computational and differential geometry, offers a powerful tool to designers to enable them to solve many of their problems in design and display of surfaces. In the present work, an attempt has been made to bridge the gap between mathematical formulations of surfaces and engineering design.

A package, SURF-3D, for design and display of three-dimensional surfaces has been developed and implemented. Design of surfaces, in this software package, is based on the spline, Bezier and b-spline formulations, atreatment of

which is given in chapter II. These schemes not only allow a designer to input and modify available surfaces, they also accommodate design and display of surfaces in a graphic interactive environment. Special routines for visualizing the design in parallel or perspective projections has been developed.

A number of design problems has been tried out. Illustrations of some of the case studies are given in chapter IV.

The salient features of the anlytical as well as the computational work are as follows:

(i) A surface generation module has been developed, which can cater to spline, Bezier, and b-spline schemes of representation. The algorithms developed are computationally fast and therefore, are useful in an interactive environment.

(ii) A editing module for creation and modification of control points has been implemented. The routines use graphical techniques to specify points.

(iii) A view module has been devised to cater to three-dimensional transformations for both parallel and perspective projections. To ease the burden of the user in specifying viewing transformations, the routines simulate a synthetic camera. In addition, independent viewing transformations like rotations

translations, and scaling have been provided.

A simple hidden surface removal algorithm has been developed. It is based on z-depth sort. It is fast enough for interactive work.

(iv) A curve module has been installed which is used for generating swept, axy-symmetric surfaces. Both, editing and the view modules can be used on these surfaces.

(v) A conversion module has been implemented for inter-conversion between surface types. This is primarily meant to ease modifications.

5.2 Suggestions for further work.

(i) The present work deals with single instances of spline, Bezier and b-spline surfaces. In practice, blending between surfaces of different types are required. It is necessary to take into account such applications.

(ii) Besides intersections of surfaces with planes, one often needs to get intersections curves of two surfaces. Modifications and/or additions to the current intersection routines, or development of a general purpose algorithm for both planes and surfaces may be required.

(iii) A very wide scope of development exists in the areas of boolean set operations like cutting, joining,

difference and intersections of surface(s). In addition, interfaces to other solid modelling systems may be thought of.

(iv) Standard exchange specifications like IGES and MAP, for transferring geometric data, can be implemented to enhance the program capabilities to pass as well as take as input, data to and from other design systems.

(v) For manufacturing applications, numerically controlled machine interfaces may be developed. The design modules may be enlarged to take care of constraints of die-sinking and die-design applications.

(vi) A more versatile hidden surface removal can be thought of. The current algorithm does not work for complex surfaces, especially those with interpenetrating areas. In addition a full-colour shaded image display algorithm may be developed to give more realism to the displays provided by the package.

REFERENCES

[1] . Barnhill, R.E. and Riesenfield, R.F., eds.
"Computer Aided Geomtric Design ".
Academic Press 1974.

[2] . Bezier, P.,
"Numerical Control : Mathematics and Applications ".
Wiley 1972

[3] . Faux, I.D. and Pratt, M.J.,
"Computational Geometry for Design and Manufacture ".
Ellis Horwood 1979.

[4] . Foley, J.D. and Van Dam, A.,
"Fundamentals of Interactive Computer Graphics ".
Addison-Wesley, 1981.

[5] . Gregory, J.A., Ed.,
"The Mathematics of Surfaces"
Oxford 1986.

[6] . Harrington, S.,
"Computer Graphics - A Programming Approach".
McGraw Hill 1983.

[7] . Mortenson, M.E.,
"Geometric Modelling".
Wiley 1985.

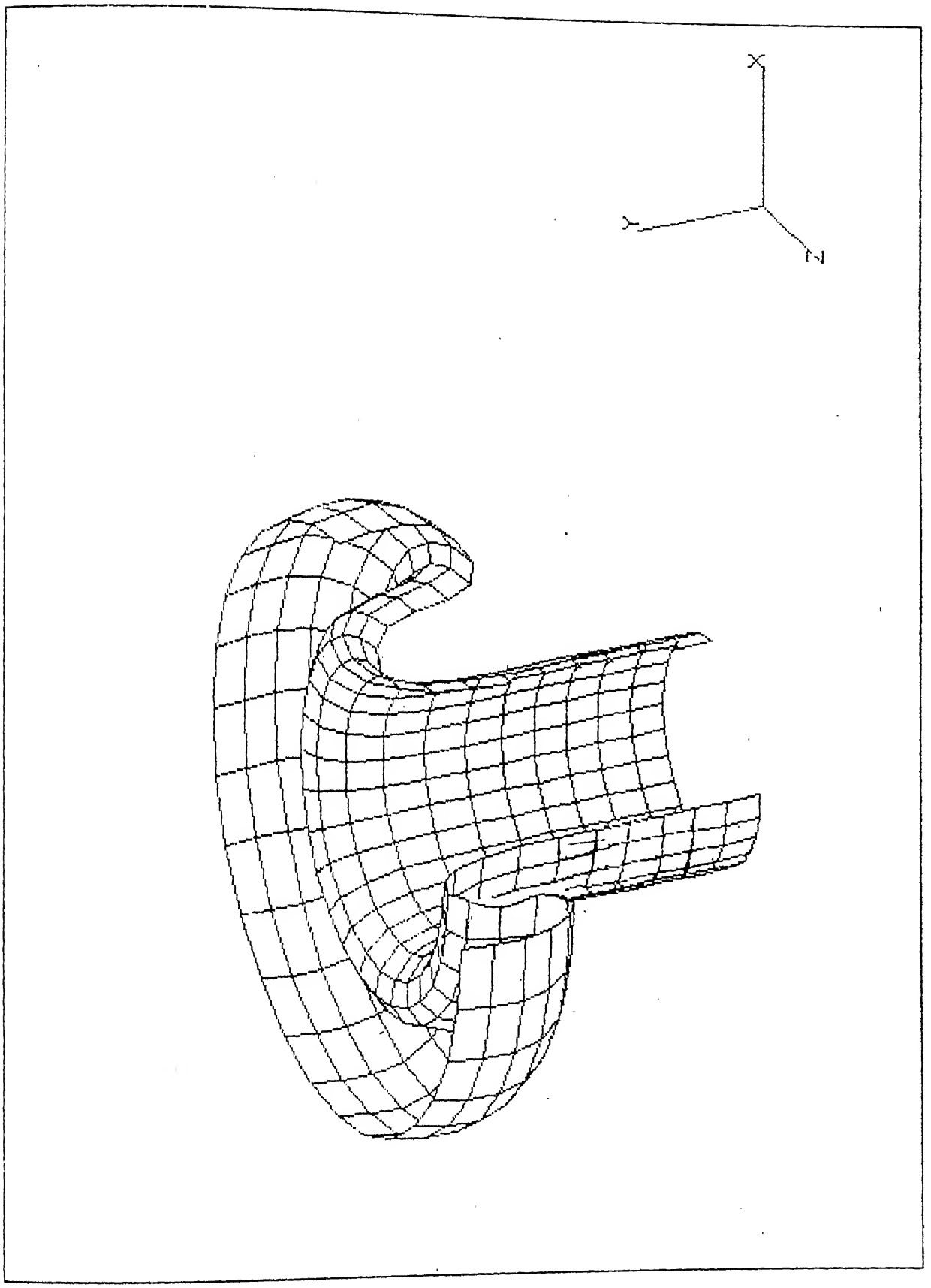
[8] . Newman, W., and Sproull, R.,
"Principles of Interactive Computer Graphics".(2nd Ed).
McGraw Hill 1979.

[9] . Rogers, D.F.,
"Procedural Elements for Computer Graphics".
Mcgraw Hill 1985.

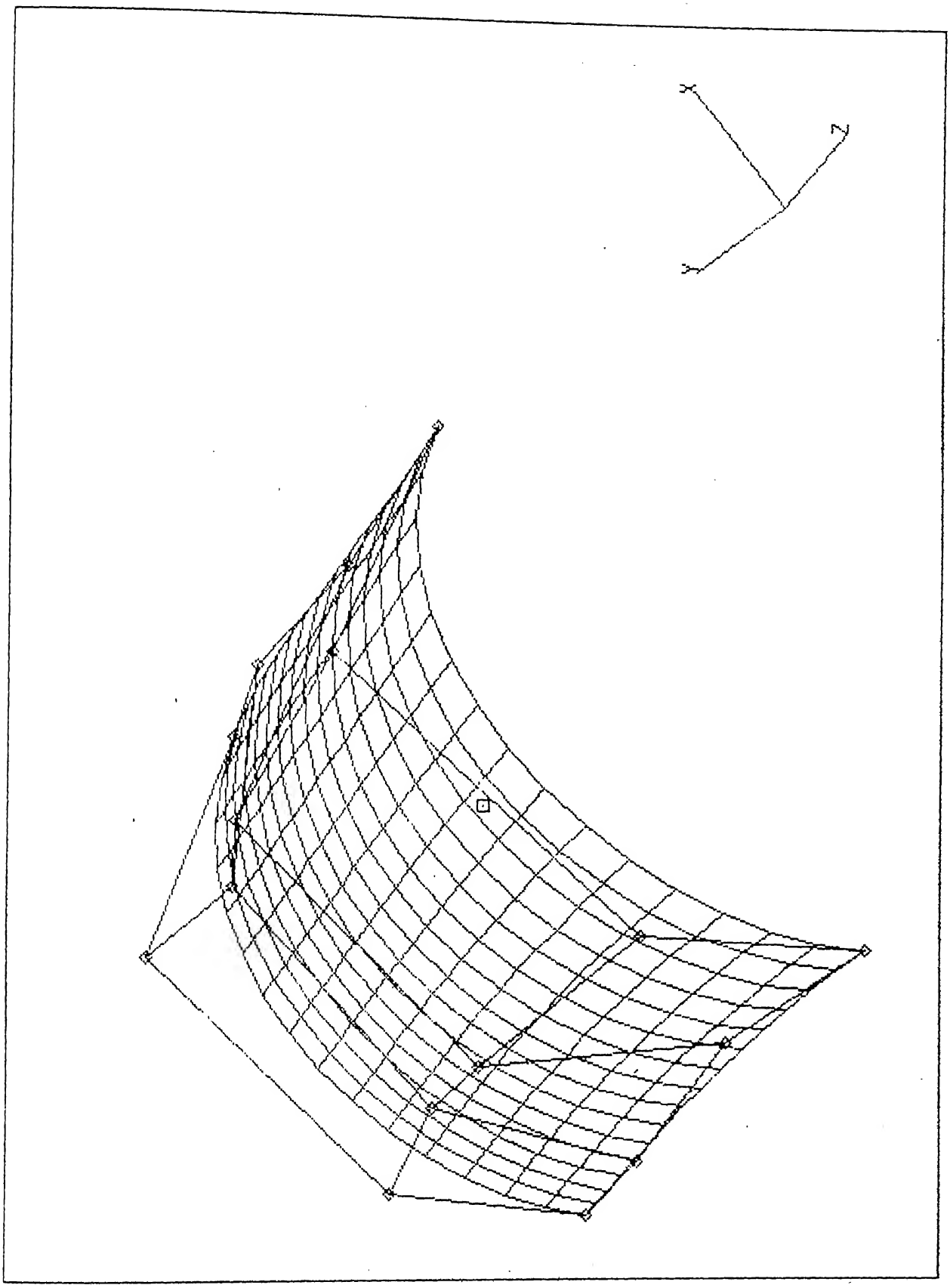
[10] . Rogers, D.F. and Adams, J.A.,
"Mathematical Elements for Computer Graphics".
McGraw Hill 1976.

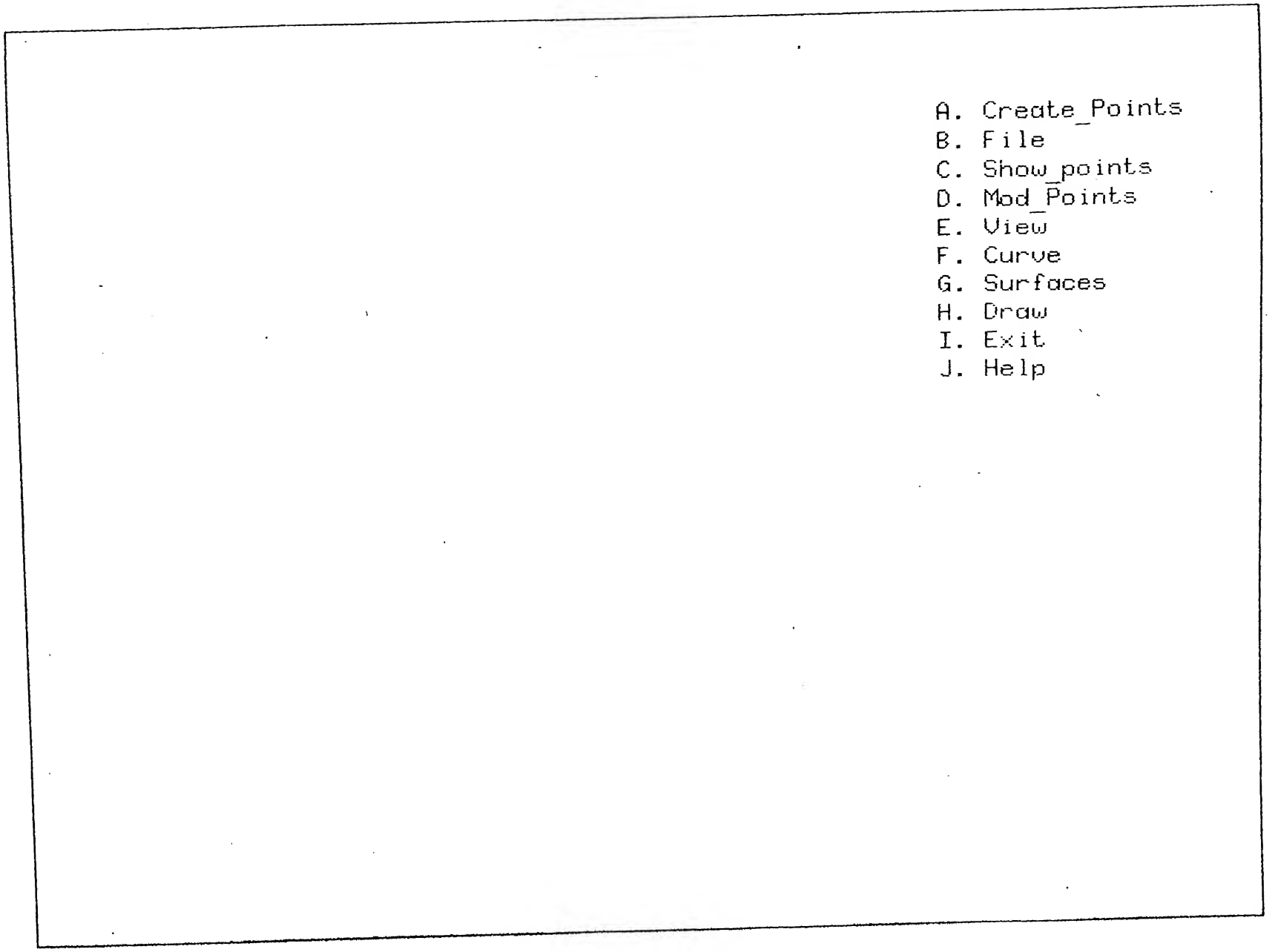
APPENDIX I

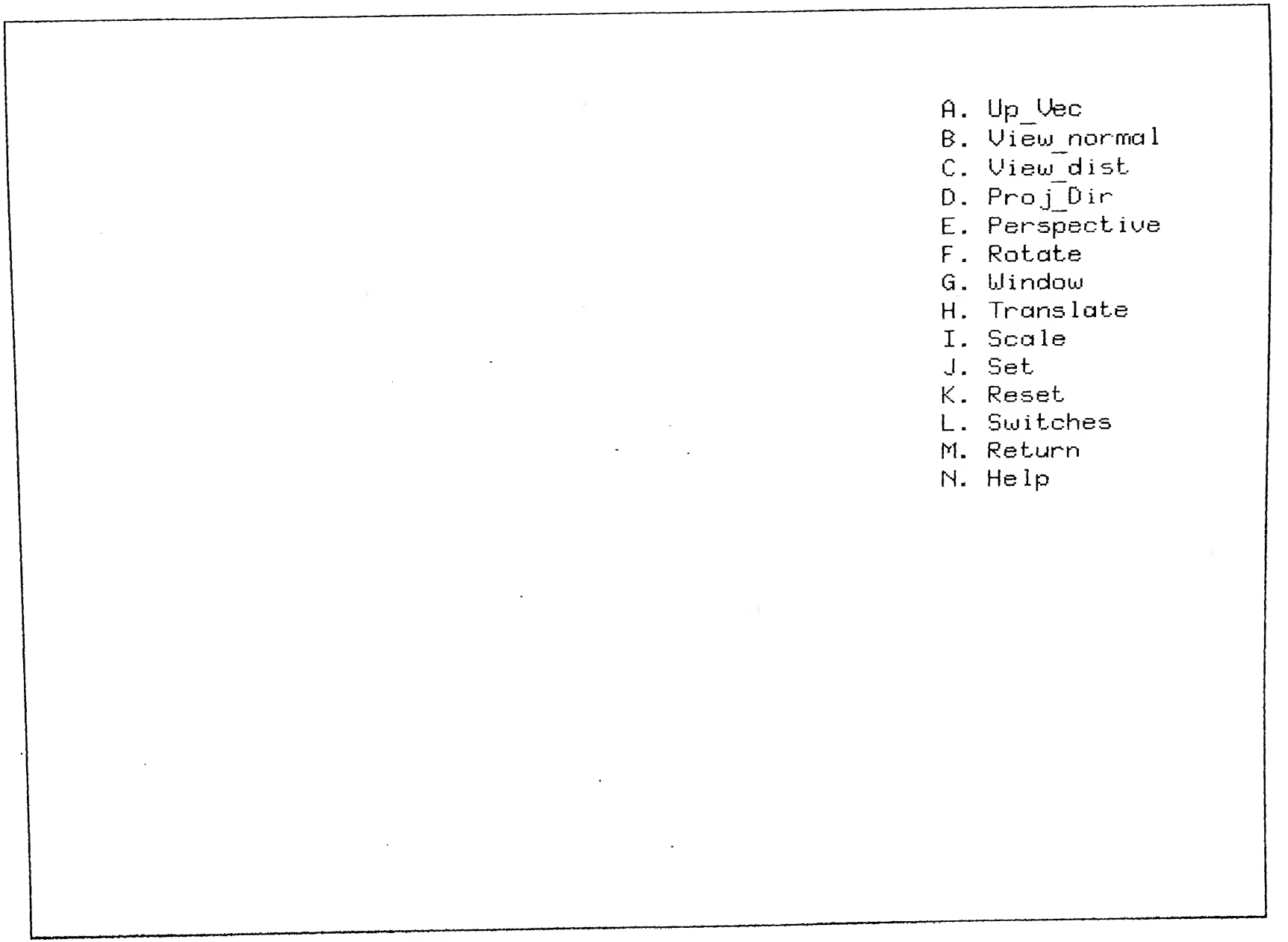
SOME SAMPLE OUTPUTS

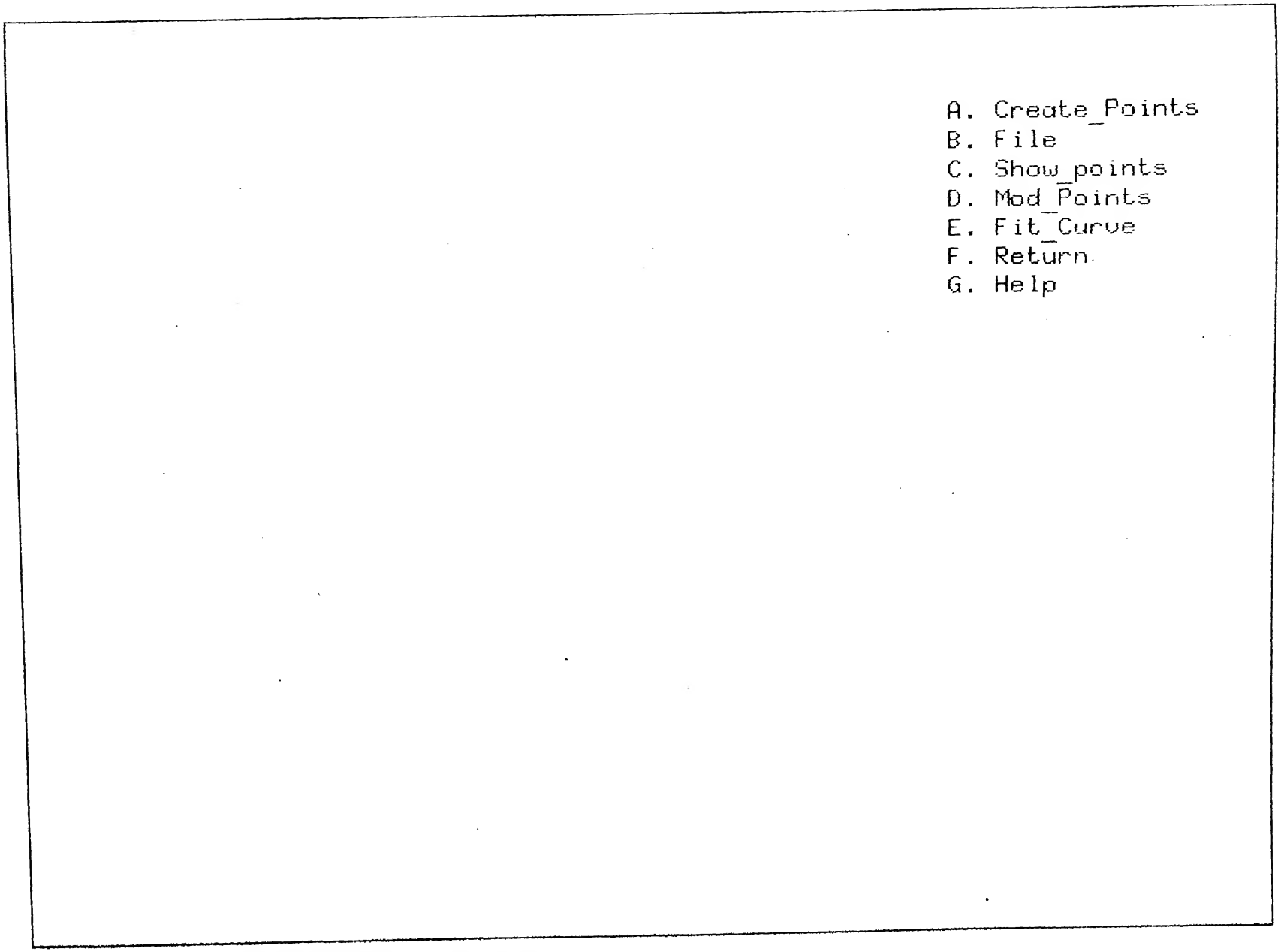

Y
Z
X

X
Z
Y

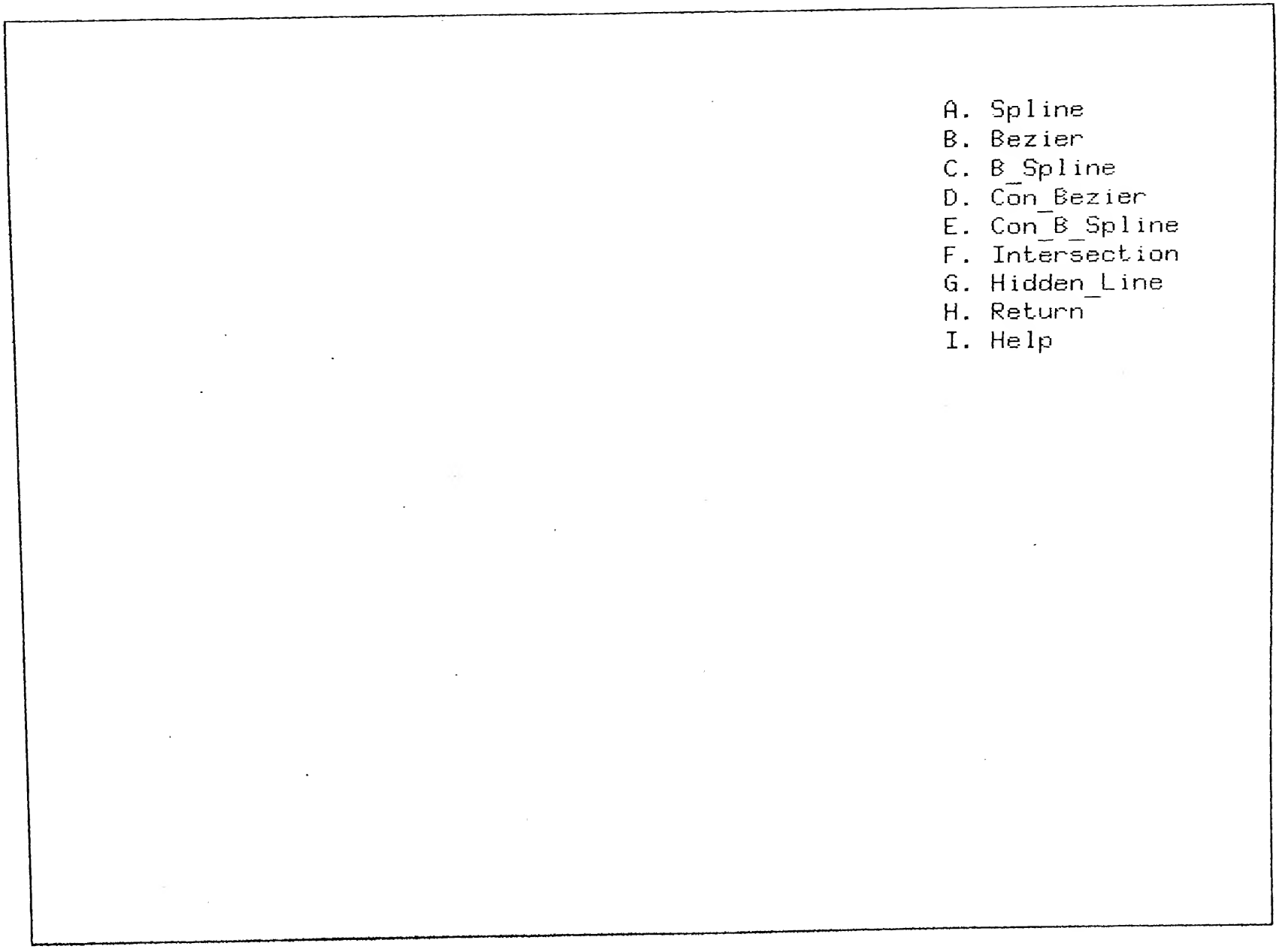

X
Z
Y

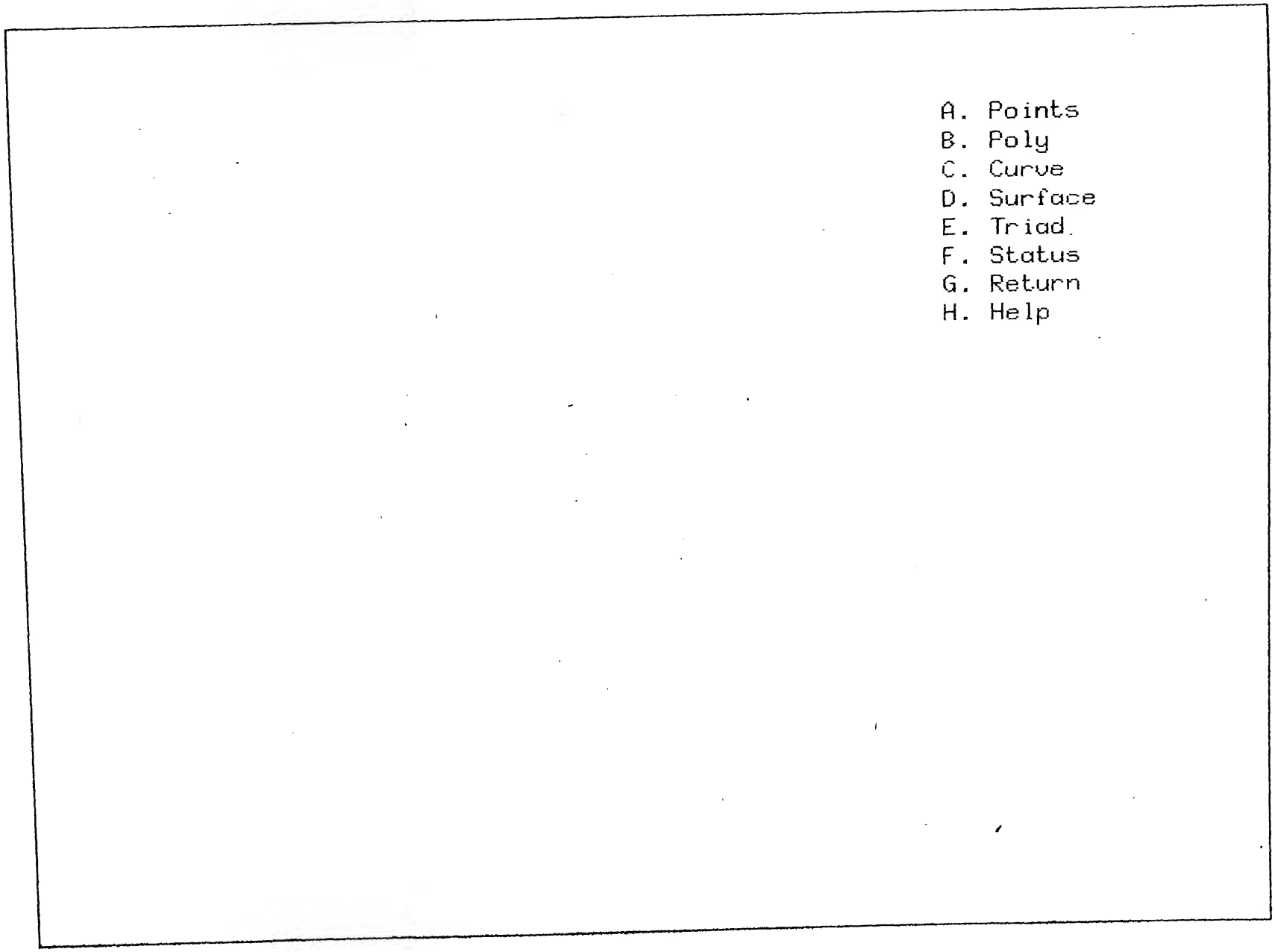

A. Create_Points
B. File
C. Show_points
D. Mod_Points
E. View
F. Curve
G. Surfaces
H. Draw
I. Exit
J. Help


A. Up_Vec
B. View_normal
C. View_dist
D. Proj_Dir
E. Perspective
F. Rotate
G. Window
H. Translate
I. Scale
J. Set
K. Reset
L. Switches
M. Return
N. Help


A. Create_Points
B. File
C. Show_points
D. Mod_Points
E. Fit_Curve
F. Return
G. Help

A. Spline
B. Bezier
C. B_Spline
D. Sweep
E. Return
F. Help


A. Spline
B. Bezier
C. B_Spline
D. Con_Bezier
E. Con_B_Spline
F. Intersection
G. Hidden_Line
H. Return
I. Help


A. Points
B. Poly
C. Curve
D. Surface
E. Triad
F. Status
G. Return
H. Help